

Rec'd

2 JUL 2004

10/541458

PCT/JP 03/16916

26.12.03

日 本 国 特 許 庁
JAPAN PATENT OFFICE

JP03/16916

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日 2 0 0 3 年 1 2 月 1 日
Date of Application:

出 願 番 号 特 願 2 0 0 3 - 4 0 2 2 0 7
Application Number:
[ST. 10/C]: [J P 2 0 0 3 - 4 0 2 2 0 7]

出 願 人 松 下 電 器 産 業 株 式 有 限 公 司
Applicant(s):

REC'D 19 FEB 2004

WIPO

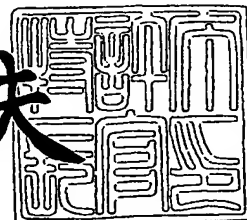
PCT

PRIORITY DOCUMENT
SUBMITTED OR TRANSMITTED IN
COMPLIANCE WITH
RULE 17.1(a) OR (b)

2 0 0 4 年 2 月 6 日

特許庁長官
Commissioner,
Japan Patent Office

今 井 康 夫



【書類名】 特許願
【整理番号】 2968150055
【提出日】 平成15年12月 1日
【あて先】 特許庁長官殿
【国際特許分類】 G06F 9/45
【発明者】
 【住所又は居所】 愛知県名古屋市中区栄2丁目6番1号 白川ビル別館5階 株式会社松下電器情報システム名古屋研究所内
 【氏名】 河合 正樹
【発明者】
 【住所又は居所】 愛知県名古屋市中区栄2丁目6番1号 白川ビル別館5階 株式会社松下電器情報システム名古屋研究所内
 【氏名】 川本 琢二
【発明者】
 【住所又は居所】 大阪府門真市大字門真1006番地 松下電器産業株式会社内
 【氏名】 春名 修介
【発明者】
 【住所又は居所】 大阪府門真市大字門真1006番地 松下電器産業株式会社内
 【氏名】 藤原 寛
【特許出願人】
 【識別番号】 000005821
 【住所又は居所】 大阪府門真市大字門真1006番地
 【氏名又は名称】 松下電器産業株式会社
【代理人】
 【識別番号】 100067828
 【弁理士】
 【氏名又は名称】 小谷 悦司
【選任した代理人】
 【識別番号】 100075409
 【弁理士】
 【氏名又は名称】 植木 久一
【選任した代理人】
 【識別番号】 100109438
 【弁理士】
 【氏名又は名称】 大月 伸介
【先の出願に基づく優先権主張】
 【出願番号】 特願2003- 275
 【出願日】 平成15年 1月 6日
【手数料の表示】
 【予納台帳番号】 012472
 【納付金額】 21,000円
【提出物件の目録】
 【物件名】 特許請求の範囲 1
 【物件名】 明細書 1
 【物件名】 図面 1
 【物件名】 要約書 1
 【包括委任状番号】 0214505

【書類名】 特許請求の範囲**【請求項 1】**

原始プログラムを目的プログラムに変換するためのコンパイラプログラムであって、
前記原始プログラムに含まれる呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するためのアドレス保存プログラムを生成するアドレス保存プログラム生成手段と、

前記呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスを設定するためのアドレス設定プログラムを生成するアドレス設定プログラム生成手段と、

前記呼び出し元プログラムモジュールに含まれる第 1 のサブプログラムから前記他プログラムモジュールに含まれる第 2 のサブプログラムへ移行するための移行プログラムを生成する移行プログラム生成手段と、

前記移行された第 2 のサブプログラムから第 1 のサブプログラムへの復帰後に、前記保存された呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するためのアドレス再設定プログラムを生成するアドレス再設定プログラム生成手段と

、
前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスする際に、前記設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムを生成するアクセスプログラム生成手段としてコンピュータを機能させることを特徴とするコンパイラプログラム。

【請求項 2】

前記原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第 1 のサブプログラムから呼び出される第 2 のサブプログラムを含む他プログラムモジュールの記述に基づいて、処理を短縮するか否かを判別する判別手段としてコンピュータをさらに機能させ、

前記判別手段によって処理を短縮しないと判別された場合、

前記アドレス保存プログラム生成手段は、前記呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するためのアドレス保存プログラムを生成し、

前記アドレス設定プログラム生成手段は、前記呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスを設定するためのアドレス設定プログラムを生成し、

前記移行プログラム生成手段は、前記呼び出し元プログラムモジュールに含まれる第 1 のサブプログラムから前記他プログラムモジュールに含まれる第 2 のサブプログラムへ移行するための移行プログラムを生成し、

前記アドレス再設定プログラム生成手段は、前記移行された第 2 のサブプログラムから第 1 のサブプログラムへの復帰後に、前記保存された呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するためのアドレス再設定プログラムを生成し、

前記アクセスプログラム生成手段は、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスする際に、前記設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムを生成し、

前記判別手段によって処理を短縮すると判別された場合、

前記移行プログラム生成手段は、前記呼び出し元プログラムモジュールに含まれる第 1 のサブプログラムから前記他プログラムモジュールに含まれる第 2 のサブプログラムへ移行するための移行プログラムを生成し、

前記アクセスプログラム生成手段は、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスする際

に、前記呼び出し元プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムを生成することを特徴とする請求項1記載のコンパイラプログラム。

【請求項3】

前記原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第1のサブプログラムから他プログラムモジュールに含まれる第2のサブプログラムが呼び出された後、

前記アドレス設定プログラム生成手段は、前記呼び出し元プログラムモジュールの実行単位に記憶している各プログラムモジュール用データメモリ領域アドレステーブルから前記他プログラムモジュール用データメモリ領域アドレスを読み出して設定するためのアドレス設定プログラムを生成し、

前記アクセスプログラム生成手段は、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスする際に、前記設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムを生成することを特徴とする請求項1記載のコンパイラプログラム。

【請求項4】

前記原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第1のサブプログラムから呼び出される第2のサブプログラムを含む他プログラムモジュールの記述に基づいて、処理を短縮するか否かを判別する判別手段としてコンピュータをさらに機能させ、

前記判別手段によって処理を短縮しないと判別された場合、

前記原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第1のサブプログラムから他プログラムモジュールに含まれる第2のサブプログラムが呼び出された後、

前記アドレス設定プログラム生成手段は、前記呼び出し元プログラムモジュールの実行単位に記憶している各プログラムモジュール用データメモリ領域アドレステーブルから前記他プログラムモジュール用データメモリ領域アドレスを読み出して設定するためのアドレス設定プログラムを生成し、

前記アクセスプログラム生成手段は、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスする際に、前記設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムを生成し、

前記判別手段によって処理を短縮すると判別された場合、

前記アクセスプログラム生成手段は、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスする際に、前記呼び出し元プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムを生成することを特徴とする請求項3記載のコンパイラプログラム。

【請求項5】

前記原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第1のサブプログラムから呼び出される第2のサブプログラムを含む他プログラムモジュールの記述は、

前記第2のサブプログラム毎に記述される宣言であることを特徴とする請求項2又は4に記載のコンパイラプログラム。

【請求項6】

前記呼び出し元プログラムモジュールによって呼び出される第2のサブプログラムを含む他プログラムモジュールを含む前記原始プログラムは、

前記他プログラムモジュールに含まれる外部変数が前記呼び出し元プログラムモジュールの複数の実行によって共通にアクセスする前記他プログラムモジュール用の実行共有デ

ータメモリ領域、及び前記他プログラムモジュールに含まれる外部変数が前記呼び出し元プログラムモジュールの実行毎に固有にアクセスする前記他プログラムモジュール用の実行固有データメモリ領域のうちのいずれを使用するのかを特定するための記述を含むことを特徴とする請求項1～5のいずれかに記載のコンパイラプログラム。

【請求項7】

前記他プログラムモジュール用の実行共有データメモリ領域、及び前記他プログラムモジュール用の実行固有データメモリ領域のうちのいずれを使用するのかを特定するための記述は、

前記他プログラムモジュールに含まれる外部変数毎に記述される宣言であることを特徴とする請求項6に記載のコンパイラプログラム。

【請求項8】

前記アドレス保存プログラム生成手段は、前記呼び出し元プログラムモジュール用の実行共有データメモリ領域アドレスを保存するとともに、前記呼び出し元プログラムモジュール用の実行固有データメモリ領域アドレスを保存するためのアドレス保存プログラムを生成し、

前記アドレス設定プログラム生成手段は、前記呼び出し元プログラムモジュールによって呼び出される他プログラムモジュール用の実行共有データメモリ領域アドレスを設定するとともに、前記呼び出し元プログラムモジュールによって呼び出される他プログラムモジュール用の実行固有データメモリ領域アドレスを設定するためのアドレス設定プログラムを生成し、

前記移行プログラム生成手段は、前記第1のサブプログラムから前記第2のサブプログラムへ移行するための移行プログラムを生成し、

前記アドレス再設定プログラム生成手段は、前記移行された第2のサブプログラムから前記第1のサブプログラムへの復帰後に、前記保存された呼び出し元プログラムモジュール用の実行固有データメモリ領域アドレスを読み出して再設定するとともに、前記保存された呼び出し元プログラムモジュール用の実行共有データメモリ領域アドレスを読み出して再設定するためのアドレス再設定プログラムを生成し、

前記アクセスプログラム生成手段は、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用の実行固有データメモリ領域をアクセスする際に、前記設定された他プログラムモジュール用の実行固有データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするとともに、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用の実行共有データメモリ領域をアクセスする際に、前記設定された他プログラムモジュール用の実行共有データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムを生成することを特徴とする請求項6に記載のコンパイラプログラム。

【請求項9】

前記原始プログラムに含まれる呼び出し元プログラムモジュールから前記他プログラムモジュールが呼び出された後、

前記アドレス設定プログラム生成手段は、前記呼び出し元プログラムモジュールの実行単位に記憶している各プログラムモジュール用データメモリ領域アドレステーブルから前記他プログラムモジュール用の実行固有データメモリ領域アドレスを読み出して設定するとともに、前記呼び出し元プログラムモジュールの複数の実行が共通にアクセスする前記他プログラムモジュール用の実行共有データメモリ領域アドレスを読み出して設定するためのアドレス設定プログラムを生成し、

前記アクセスプログラム生成手段は、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用の実行固有データメモリ領域をアクセスする際に、前記設定された他プログラム用の実行固有データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするとともに、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用の実行共有

データメモリ領域をアクセスする際に、前記設定された他プログラムモジュール用の実行共有データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムを生成することを特徴とする請求項6記載のコンパイラプログラム。

【請求項10】

前記アドレス保存プログラム生成手段は、前記第1のサブプログラムから第2のサブプログラムへ移行するために必要なデータよりも先に、前記呼び出し元プログラムモジュール用の実行共有データメモリ領域アドレス、及び実行固有データメモリ領域アドレスをスタックメモリに保存するためのアドレス保存プログラムを生成することを特徴とする請求項1に記載のコンパイラプログラム。

【請求項11】

前記原始プログラムに含まれる前記呼び出し元プログラムモジュールと、

前記呼び出し元プログラムモジュールに含まれる第1のサブプログラムから呼び出される第2のサブプログラムを含む他プログラムモジュールとは、同じプログラムモジュールであることを特徴とする請求項1～10のいずれかに記載のコンパイラプログラム。

【請求項12】

前記呼び出し元プログラムモジュールはコード領域とデータ領域とを含み、前記他プログラムモジュールはコード領域とデータ領域とを含み、

前記呼び出し元プログラムモジュールから前記他プログラムモジュールへの呼び出し命令を受けて、前記他プログラムモジュールを識別する識別番号を取得するための識別番号取得プログラムを生成する識別番号取得プログラム生成手段と、

前記取得された識別番号をインデックスとして、前記呼び出し元プログラムモジュールの識別番号と、前記呼び出し元プログラムモジュールのデータ領域の先頭アドレスとが対応付けられるとともに、前記他プログラムモジュールの識別番号と、前記他プログラムモジュールのデータ領域の先頭アドレスとが対応付けられたテーブルから前記他プログラムモジュールのデータ領域の先頭アドレスを取得するための先頭アドレス取得プログラムを生成する先頭アドレス取得プログラム生成手段と、

前記取得された前記他プログラムモジュールのデータ領域の先頭アドレスと、前記呼び出し元プログラムモジュールのデータ領域の先頭アドレスとを切り替えるための先頭アドレス切替プログラムを生成するための先頭アドレス切替プログラム生成手段としてコンピュータをさらに機能させることを特徴とする請求項1記載のコンパイラプログラム。

【請求項13】

前記呼び出し元プログラムモジュールはコード領域とデータ領域とを含み、前記他プログラムモジュールはコード領域とデータ領域とを含み、

前記呼び出し元プログラムモジュールを識別する識別番号と、前記他プログラムモジュールを識別する識別番号とをそれぞれに付与するための識別番号付与プログラムを生成する識別番号付与プログラム生成手段と、

前記付与された前記呼び出し元プログラムモジュールの識別番号と、前記呼び出し元プログラムモジュールのデータ領域の先頭アドレスとを対応付けるとともに、前記付与された前記他プログラムモジュールの識別番号と、前記他プログラムモジュールのデータ領域の先頭アドレスとを対応付けるテーブルを作成するためのテーブル作成プログラムを生成するテーブル作成プログラム生成手段と、

前記呼び出し元プログラムモジュールから前記他プログラムモジュールへの呼び出し命令を受けて、前記他プログラムモジュールの識別番号を取得するための識別番号取得プログラムを生成する識別番号取得プログラム生成手段と、

前記取得された識別番号をインデックスとして前記作成されたテーブルから前記他プログラムモジュールのデータ領域の先頭アドレスを取得するための先頭アドレス取得プログラムを生成する先頭アドレス取得プログラム生成手段と、

前記取得された前記他プログラムモジュールのデータ領域の先頭アドレスと、前記呼び出し元プログラムモジュールのデータ領域の先頭アドレスとを切り替えるための先頭アド

レス切替プログラムを生成するための先頭アドレス切替プログラム生成手段としてコンピュータをさらに機能させることを特徴とする請求項1記載のコンパイラプログラム。

【請求項14】

前記他プログラムモジュール外で定義された変数又は関数へのアドレスを保持するためのアドレス保持プログラムを生成するアドレス保持プログラム生成手段としてコンピュータをさらに機能させ、

前記アクセスプログラム生成手段は、前記他プログラムモジュール内で定義された変数へアクセスする場合、前記取得された先頭アドレスからの相対アドレスでアクセスし、前記他プログラムモジュール内で定義された関数へアクセスする場合、プログラムカウンタからの相対アドレスでアクセスし、前記他プログラムモジュール外で定義された変数又は関数へアクセスする場合、前記保持されたアドレスに前記取得された先頭アドレスからの相対アドレスでアクセスした後、前記保持されたアドレス経由で間接アクセスするためのアクセスプログラムを生成することを特徴とする請求項1記載のコンパイラプログラム。

【請求項15】

少なくとも1つのモジュールで構成され、かつ現在実行中であるアプリケーションを終了するための終了プログラムを生成する終了プログラム生成手段と、

アプリケーションが終了された後、メモリをコンパクションするためのコンパクションプログラムを生成するコンパクションプログラム生成手段と、

メモリがコンパクションされた後、前記アプリケーションを最初から再開するための再開プログラムを生成する再開プログラム生成手段としてコンピュータをさらに機能させることを特徴とする請求項1記載のコンパイラプログラム。

【請求項16】

少なくとも1つのモジュールで構成され、かつ現在実行中であるアプリケーションを終了するための終了プログラムを生成する終了プログラム生成手段と、

終了されるまでの前記アプリケーションのアドレス値に依存しない情報を保存するための保存プログラムを生成する保存プログラム生成手段と、

メモリをコンパクションするためのコンパクションプログラムを生成するコンパクションプログラム生成手段と、

メモリがコンパクションされた後、前記保存された情報を読み出し、読み出された情報に基づいて前記アプリケーションが終了した時点における状態まで当該アプリケーションを実行するための実行プログラムを生成する実行プログラム生成手段としてコンピュータをさらに機能させることを特徴とする請求項1記載のコンパイラプログラム。

【請求項17】

メモリ内において、アドレス値を設定するか否かを示す識別子を設け、

コンパクション前のメモリの状態を記憶するためのメモリ状態記憶プログラムを生成するメモリ状態記憶プログラム生成手段と、

メモリをコンパクションするためのコンパクションプログラムを生成するコンパクションプログラム生成手段と、

メモリがコンパクションされた後、前記識別子に基づいてアドレス値が設定されているエントリに対して、前記記憶されているコンパクション前のメモリの状態を参照して、アドレス値の再配置を行うための再配置プログラムを生成する再配置プログラム生成手段としてコンピュータをさらに機能させることを特徴とする請求項1記載のコンパイラプログラム。

【請求項18】

アドレス値を格納するアドレス値用メモリと、アドレス値以外を格納する非アドレス値用メモリとを設け、

コンパクション前のアドレス値用メモリ及び非アドレス値用メモリの状態を記憶するためのメモリ状態記憶プログラムを生成するメモリ状態記憶プログラム生成手段と、

前記アドレス値用メモリ及び前記非アドレス値用メモリをコンパクションするためのコンパクションプログラムを生成するコンパクションプログラム生成手段と、

前記アドレス値用メモリ及び前記非アドレス値用メモリがコンパクションされた後、前記記憶されているコンパクション前のアドレス値用メモリの状態を参照して、前記アドレス値用メモリのみのアドレス値の再配置を行うための再配置プログラムを生成する再配置プログラム生成手段としてコンピュータをさらに機能させることを特徴とする請求項1記載のコンパイラプログラム。

【請求項19】

メモリ内において、モジュールを識別するための識別番号と、前記モジュールのデータ領域へのアドレスを設定するか、コード領域へのアドレスを設定するかを識別する識別子とを設け、

ポインタにアドレス値を代入する際に前記識別番号と前記識別子とを設定するための設定プログラムを生成する設定プログラム生成手段と、

前記メモリをコンパクションするためのコンパクションプログラムを生成するコンパクションプログラム生成手段と、

前記メモリがコンパクションされた後、前記識別番号及び前記識別子に基づいてアドレス値を再計算するための再計算プログラムを生成する再計算プログラム生成手段としてコンピュータをさらに機能させることを特徴とする請求項1記載のコンパイラプログラム。

【請求項20】

メモリ内において、モジュールを識別するための識別番号と、前記モジュールのデータ領域へのアドレスを設定するか、コード領域へのアドレスを設定するかを識別する識別子とを設け、

ポインタにアドレス値を代入する際に前記識別番号と前記識別子とを設定するとともに、データ領域の先頭アドレス又はコード領域の先頭アドレスからのオフセット値を設定するためのオフセット値設定プログラムを生成するオフセット値設定プログラム生成手段と

、
前記ポインタを使用する際に、前記オフセット値にデータ領域の先頭アドレス又はコード領域の先頭アドレスを加算することによって実アドレスに変換するための実アドレス変換プログラムを生成する実アドレス変換プログラム生成手段としてコンピュータをさらに機能させることを特徴とする請求項1記載のコンパイラプログラム。

【請求項21】

モジュールのコード領域又はデータ領域毎にコンパクションするためのコンパクションプログラムを生成するコンパクションプログラム生成手段と、

コンパクションによって移動する前記コード領域又は前記データ領域を指すハンドルがあるか否かを検索するための検索プログラムを生成する検索プログラム生成手段と、

前記コード領域又は前記データ領域を指すハンドルが発見された場合、当該ハンドルを再配置するためのハンドル再配置プログラムを生成するハンドル再配置プログラム生成手段としてコンピュータをさらに機能させ、

前記コンパクションプログラム生成手段は、全てのハンドルについて再配置を行った後、当該コード領域又はデータ領域をコンパクションするためのコンパクションプログラムを生成することを特徴とする請求項1記載のコンパイラプログラム。

【請求項22】

原始プログラムを目的プログラムに変換するためのコンパイラプログラムを記録したコンピュータ読み取り可能な記録媒体であって、

前記原始プログラムに含まれる呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するためのアドレス保存プログラムを生成するアドレス保存プログラム生成手段と、

前記呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスを設定するためのアドレス設定プログラムを生成するアドレス設定プログラム生成手段と、

前記呼び出し元プログラムモジュールに含まれる第1のサブプログラムから前記他プログラムモジュールに含まれる第2のサブプログラムへ移行するための移行プログラムを生

成する移行プログラム生成手段と、

前記移行された第2のサブプログラムから第1のサブプログラムへの復帰後に、前記保存された呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するためのアドレス再設定プログラムを生成するアドレス再設定プログラム生成手段と

、
前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスする際に、前記設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムを生成するアクセスプログラム生成手段としてコンピュータを機能させることを特徴とするコンパイラプログラムを記録したコンピュータ読み取り可能な記録媒体。

【請求項23】

原始プログラムを目的プログラムに変換するためのコンパイル方法であって、

コンピュータが、前記原始プログラムに含まれる呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するためのアドレス保存プログラムを生成するアドレス保存プログラム生成ステップと、

コンピュータが、前記呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスを設定するためのアドレス設定プログラムを生成するアドレス設定プログラム生成ステップと、

コンピュータが、前記呼び出し元プログラムモジュールに含まれる第1のサブプログラムから前記他プログラムモジュールに含まれる第2のサブプログラムへ移行するための移行プログラムを生成する移行プログラム生成ステップと、

コンピュータが、前記移行された第2のサブプログラムから第1のサブプログラムへの復帰後に、前記保存された呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するためのアドレス再設定プログラムを生成するアドレス再設定プログラム生成ステップと、

コンピュータが、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスする際に、前記設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムを生成するアクセスプログラム生成ステップを含むことを特徴とするコンパイル方法。

【請求項24】

原始プログラムを目的プログラムに変換するコンパイル装置であって、

前記原始プログラムに含まれる呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するためのアドレス保存プログラムを生成するアドレス保存プログラム生成手段と、

前記呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスを設定するためのアドレス設定プログラムを生成するアドレス設定プログラム生成手段と、

前記呼び出し元プログラムモジュールに含まれる第1のサブプログラムから前記他プログラムモジュールに含まれる第2のサブプログラムへ移行するための移行プログラムを生成する移行プログラム生成手段と、

前記移行された第2のサブプログラムから第1のサブプログラムへの復帰後に、前記保存された呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するためのアドレス再設定プログラムを生成するアドレス再設定プログラム生成手段と

、
前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスする際に、前記設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムを生成するアクセスプログラム生成手段とを備えることを

特徴とするコンパイル装置。

【請求項 25】

原始プログラムが変換されることによって生成される目的プログラムであって、

前記原始プログラムに含まれる呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するアドレス保存手段と、

前記呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスを設定するアドレス設定手段と、

前記呼び出し元プログラムモジュールに含まれる第1のサブプログラムから前記他プログラムモジュールに含まれる第2のサブプログラムへ移行する移行手段と、

前記移行手段によって移行された第2のサブプログラムから第1のサブプログラムへの復帰後に、前記アドレス保存手段によって保存された呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するアドレス再設定手段と、

前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスする際に、前記アドレス設定手段によって設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするアクセス手段としてコンピュータを機能させることを特徴とする目的プログラム。

【請求項 26】

原始プログラムが変換されることによって生成される目的プログラムを記録したコンピュータ読み取り可能な記録媒体であって、

前記原始プログラムに含まれる呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するアドレス保存手段と、

前記呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスを設定するアドレス設定手段と、

前記呼び出し元プログラムモジュールに含まれる第1のサブプログラムから前記他プログラムモジュールに含まれる第2のサブプログラムへ移行する移行手段と、

前記移行手段によって移行された第2のサブプログラムから第1のサブプログラムへの復帰後に、前記アドレス保存手段によって保存された呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するアドレス再設定手段と、

前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスする際に、前記アドレス設定手段によって設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするアクセス手段としてコンピュータを機能させることを特徴とする目的プログラムを記録したコンピュータ読み取り可能な記録媒体。

【請求項 27】

原始プログラムが変換されることによって生成される目的プログラムを実行する目的プログラム実行方法であって、

コンピュータが、前記原始プログラムに含まれる呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するアドレス保存ステップと、

コンピュータが、前記呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスを設定するアドレス設定ステップと、

コンピュータが、前記呼び出し元プログラムモジュールに含まれる第1のサブプログラムから前記他プログラムモジュールに含まれる第2のサブプログラムへ移行する移行ステップと、

コンピュータが、前記移行ステップにおいて移行された第2のサブプログラムから第1のサブプログラムへの復帰後に、前記アドレス保存ステップにおいて保存された呼び出し

元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するアドレス再設定ステップと、

コンピュータが、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスする際に、前記アドレス設定ステップにおいて設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするアクセスステップとを含むことを特徴とする目的プログラム実行方法。

【請求項 28】

原始プログラムが変換されることによって生成される目的プログラムを実行する目的プログラム実行装置であって、

前記原始プログラムに含まれる呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するアドレス保存手段と、

前記呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスを設定するアドレス設定手段と、

前記呼び出し元プログラムモジュールに含まれる第1のサブプログラムから前記他プログラムモジュールに含まれる第2のサブプログラムへ移行する移行手段と、

前記移行手段によって移行された第2のサブプログラムから第1のサブプログラムへの復帰後に、前記アドレス保存手段によって保存された呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するアドレス再設定手段と、

前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスする際に、前記アドレス設定手段によって設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするアクセス手段とを備えることを特徴とする目的プログラム実行装置。

【請求項 29】

前記呼び出し元プログラムモジュールはコード領域とデータ領域とを含み、前記他プログラムモジュールはコード領域とデータ領域とを含み、

前記呼び出し元プログラムモジュールを識別する識別番号と、前記他プログラムモジュールを識別する識別番号とをそれぞれに付与する識別番号付与手段と、

前記識別番号付与手段によって付与された前記呼び出し元プログラムモジュールの識別番号と、前記呼び出し元プログラムモジュールのデータ領域の先頭アドレスとを対応付けるとともに、前記識別番号付与手段によって付与された前記他プログラムモジュールの識別番号と、前記他プログラムモジュールのデータ領域の先頭アドレスとを対応付けるテーブルを作成するテーブル作成手段と、

前記呼び出し元プログラムモジュールから前記他プログラムモジュールへの呼び出し命令を受けて、前記他プログラムモジュールの識別番号を取得する識別番号取得手段と、

前記識別番号取得手段によって取得された識別番号をインデクスとして、前記テーブル作成手段によって作成されたテーブルから前記他プログラムモジュールのデータ領域の先頭アドレスを取得する先頭アドレス取得手段と、

前記先頭アドレス取得手段によって取得された前記他プログラムモジュールのデータ領域の先頭アドレスと、前記呼び出し元プログラムモジュールのデータ領域の先頭アドレスとを切り替える先頭アドレス切替手段とをさらに備えることを特徴とする請求項 28 記載の目的プログラム実行装置。

【請求項 30】

前記呼び出し元プログラムモジュールはコード領域とデータ領域とを含み、前記他プログラムモジュールはコード領域とデータ領域とを含み、

前記呼び出し元プログラムモジュールを識別する識別番号と、前記他プログラムモジュールを識別する識別番号とをそれぞれに付与する識別番号付与手段と、

前記識別番号付与手段によって付与された前記呼び出し元プログラムモジュールの識別

番号と、前記呼び出し元プログラムモジュールのデータ領域の先頭アドレスとを対応付けるとともに、前記識別番号付与手段によって付与された前記他プログラムモジュールの識別番号と、前記他プログラムモジュールのデータ領域の先頭アドレスとを対応付けるテーブルを作成するテーブル作成手段とをさらに備えることを特徴とする請求項 28 記載の目的プログラム実行装置。

【書類名】明細書

【発明の名称】コンパイラプログラム、コンパイラプログラムを記録したコンピュータ読み取り可能な記録媒体、コンパイル方法及びコンパイル装置

【技術分野】

【0001】

本発明は、あるプログラム言語で記述された原始プログラム（以後、「ソースプログラム」または「ソースプログラムコード」または「ソースコード」とも言う）を、あるコンピュータによって実行するための目的プログラム（「以後、オブジェクトプログラム」または「オブジェクトコードプログラム」または「オブジェクトコード」とも言う）に変換（以後場合によっては「翻訳」と言うこともあるが同一内容を示す）するためのコンパイラプログラム、コンパイラプログラムを記録したコンピュータ読み取り可能な記録媒体、コンパイル方法及びコンパイル装置に関するものである。

【0002】

また本特許出願の明細書で言う、目的プログラムまたはオブジェクトプログラム等には、コンピュータが直接実行可能なプログラムコードに加え、コンピュータが直接実行可能なプログラムコードに変換する1段階である中間プログラムコードをも含むものであり、またそれに加え特に、コンピュータが直接実行可能なプログラムコードでは確定すべき実際にアクセスされるメモリアドレスが確定していない段階の中間コードプログラムをも含むものであり、更にまたそれに加え特に、コンピュータプログラムが直接実行可能なプログラムコード全体を一体化する以前の段階の中間コードプログラムをも含むものである。

【0003】

また例えば、いわゆる機器組み込みプログラムのようにオブジェクトプログラムを格納することのできるメモリ領域が限られており、しかも近年の携帯情報機器のようにその限られたメモリ領域に高度且つ高機能な情報処理のためのオブジェクトプログラムを格納しなければならないような目的に適う、オブジェクトプログラムを生成するためのコンパイラプログラム、コンパイラプログラムを記録したコンピュータ読み取り可能な記録媒体、コンパイル方法及びコンパイル装置に関するものである。

【0004】

更にまた、このような目的に適うオブジェクトプログラムとして再入可能（以後、「リエントラント：reentrant」と言うこともある）なオブジェクトプログラムを自動生成するためのコンパイラプログラム、コンパイラプログラムを記録したコンピュータ読み取り可能な記録媒体、コンパイル方法及びコンパイル装置に関するものである。

【背景技術】

【0005】

従来、時分割または優先度プリエンプティブタスクまたは、プロセススケジューリング機能、及びそれに類する機能を有するオペレーティングシステムや、関数型プログラミング言語を用いることを条件として制作されるプログラムであって、複数のタスクまたはプロセスからある一定の時間間隔の中で、同時に呼び出される可能性がある関数プログラムは再入可能であることが要求された。

【0006】

そしてこのような再入可能関数プログラムは、人間のプログラマによってコーディングされ、再入可能性が満足されているか否かを自動的に判定するためのプログラムが作成されていた（例えば、特許文献1参照）。

【0007】

この再入可能性を満足する条件は、判定を行う対象である関数を記述しているソースプログラムの全体を構文解析してその全フローを抽出し、その全フローでアクセスされるすべてのリソースに対して、スコープを決定し、そのリソースのスコープの種類に応じて完全な排他的制御が行われているか否かに依存するものであった。

【特許文献1】特開2001-134431号公報

【発明の開示】

【発明が解決しようとする課題】

【0008】

しかしながら、ソースプログラムの全体を構文解析してその全フローを抽出し、その全フローでアクセスされるすべてのリソースに対して、スコープを決定し、そのリソースのスコープの種類に応じて完全な排他的制御が行われているか否かを完全に行うことは時間的な制約やコンピュータの処理能力の限界から、現実には非常に困難で、ある一定レベルの判定に留まらざるを得なかった。

【0009】

そして、従来の再入可能なプログラムの作成は、その本質的部分が人間のプログラマの作業によって記述されるため、多くの時間を要しながら不完全な排他的制御しか実現することができず、その結果、プログラムを実行させた時に多くの誤動作を引き起こす原因となっていた。

【0010】

また、このような再入可能なプログラムはマルチタスクや時分割環境下で使用されることが多いため、その原因発見は極めて困難であった。更にまた、このような再入可能なプログラムの更新や改訂を行うことは、更に一層困難なものであった。

【0011】

本発明は、上記の課題を解決するためになされたもので、自動的に再入可能な目的プログラムを生成することができるコンパイラプログラム、コンパイラプログラムを記録したコンピュータ読み取り可能な記録媒体、コンパイル方法及びコンパイル装置を提供することを目的とするものである。

【課題を解決するための手段】

【0012】

本発明に係るコンパイラプログラムは、原始プログラムを目的プログラムに変換するためのコンパイラプログラムであって、前記原始プログラムに含まれる呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するためのアドレス保存プログラムを生成するアドレス保存プログラム生成手段と、前記呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスを設定するためのアドレス設定プログラムを生成するアドレス設定プログラム生成手段と、前記呼び出し元プログラムモジュールに含まれる第1のサブプログラムから前記他プログラムモジュールに含まれる第2のサブプログラムへ移行するための移行プログラムを生成する移行プログラム生成手段と、前記移行された第2のサブプログラムから第1のサブプログラムへの復帰後に、前記保存された呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するためのアドレス再設定プログラムを生成するアドレス再設定プログラム生成手段と、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスする際に、前記設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムを生成するアクセスプログラム生成手段としてコンピュータを機能させる。

【0013】

この構成によれば、アドレス保存プログラム生成手段によって、原始プログラムに含まれる呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するためのアドレス保存プログラムが生成される。そして、アドレス設定プログラム生成手段によって、呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスを設定するためのアドレス設定プログラムが生成される。さらに、移行プログラム生成手段によって、呼び出し元プログラムモジュールに含まれる第1のサブプログラムから他プログラムモジュールに含まれる第2のサブプログラムへ移行するための移行プ

プログラムが生成される。アドレス再設定プログラム生成手段によって、移行された第2のサブプログラムから第1のサブプログラムへの復帰後に、保存された呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するためのアドレス再設定プログラムが生成される。アクセスプログラム生成手段によって、他プログラムモジュールが他プログラムモジュールに含まれる他プログラムモジュール用データメモリ領域をアクセスする際に、設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムが生成される。

【0014】

したがって、他プログラムモジュールに対して呼び出し元プログラムモジュールによる再入が発生したとしても、呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスが保存されるので、呼び出し元プログラムモジュール用データメモリ領域アドレスを退避させることができ、呼び出し元プログラムモジュールによって呼び出される他プログラムモジュール用データメモリ領域アドレスが設定されるので、再入可能な目的プログラムを生成することができ、モジュール間の動的リンクを実現することができる。

【0015】

また、上記のコンパイラプログラムにおいて、前記原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第1のサブプログラムから呼び出される第2のサブプログラムを含む他プログラムモジュールの記述に基づいて、処理を短縮するか否かを判別する判別手段としてコンピュータをさらに機能させ、前記判別手段によって処理を短縮しないと判別された場合、前記アドレス保存プログラム生成手段は、前記呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するためのアドレス保存プログラムを生成し、前記アドレス設定プログラム生成手段は、前記呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスを設定するためのアドレス設定プログラムを生成し、前記移行プログラム生成手段は、前記呼び出し元プログラムモジュールに含まれる第1のサブプログラムから前記他プログラムモジュールに含まれる第2のサブプログラムへ移行するための移行プログラムを生成し、前記アドレス再設定プログラム生成手段は、前記移行された第2のサブプログラムから第1のサブプログラムへの復帰後に、前記保存された呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するためのアドレス再設定プログラムを生成し、前記アクセスプログラム生成手段は、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスする際に、前記設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムを生成し、前記判別手段によって処理を短縮すると判別された場合、前記移行プログラム生成手段は、前記呼び出し元プログラムモジュールに含まれる第1のサブプログラムから前記他プログラムモジュールに含まれる第2のサブプログラムへ移行するための移行プログラムを生成し、前記アクセスプログラム生成手段は、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスする際に、前記呼び出し元プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムを生成することが好ましい。

【0016】

この構成によれば、判別手段によって、原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第1のサブプログラムから呼び出される第2のサブプログラムを含む他プログラムモジュールの記述に基づいて、処理を短縮するか否かが判別される。判別手段によって処理を短縮しないと判別された場合、アドレス保存プログラム生成手段によって、呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するためのアドレス保存プログラムが生成される。そし

て、アドレス設定プログラム生成手段によって、呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスを設定するためのアドレス設定プログラムが生成される。さらに、移行プログラム生成手段によって、呼び出し元プログラムモジュールに含まれる第1のサブプログラムから他プログラムモジュールに含まれる第2のサブプログラムへ移行するための移行プログラムが生成される。また、アドレス再設定プログラム生成手段によって、移行された第2のサブプログラムから第1のサブプログラムへの復帰後に、保存された呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するためのアドレス再設定プログラムが生成される。さらにまた、アクセスプログラム生成手段によって、他プログラムモジュールが他プログラムモジュールに含まれる他プログラムモジュール用データメモリ領域をアクセスする際に、設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムが生成される。

【0017】

一方、判別手段によって処理を短縮すると判別された場合、移行プログラム生成手段によって、呼び出し元プログラムモジュールに含まれる第1のサブプログラムから他プログラムモジュールに含まれる第2のサブプログラムへ移行するための移行プログラムが生成される。そして、アクセスプログラム生成手段によって、他プログラムモジュールが他プログラムモジュールに含まれる他プログラムモジュール用データメモリ領域をアクセスする際に、呼び出し元プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムが生成される。

【0018】

したがって、呼び出し元プログラムモジュールに含まれる第1のサブプログラムから呼び出される第2のサブプログラムを含む他プログラムモジュールの記述に基づいて、処理を短縮するか否かが判別されるので、必ずしも呼び出し元プログラムモジュールと引き出される他プログラムモジュールとが異なるデータ領域を使用することが必要でないケースでは、アドレスの保存や設定や再設定の処理を省略し、目的プログラムの実行を高速化することができ、不要な処理を出力しないことで目的プログラムのサイズを小さくすることができる。

【0019】

また、上記のコンパイラプログラムにおいて、前記原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第1のサブプログラムから他プログラムモジュールに含まれる第2のサブプログラムが呼び出された後、前記アドレス設定プログラム生成手段は、前記呼び出し元プログラムモジュールの実行単位に記憶している各プログラムモジュール用データメモリ領域アドレステーブルから前記他プログラムモジュール用データメモリ領域アドレスを読み出して設定するためのアドレス設定プログラムを生成し、前記アクセスプログラム生成手段は、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスする際に、前記設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムを生成することが好ましい。

【0020】

この構成によれば、原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第1のサブプログラムから他プログラムモジュールに含まれる第2のサブプログラムが呼び出された後、アドレス設定プログラム生成手段によって、呼び出し元プログラムモジュールの実行単位に記憶している各プログラムモジュール用データメモリ領域アドレステーブルから他プログラムモジュール用データメモリ領域アドレスを読み出して設定するためのアドレス設定プログラムが生成される。そして、アクセスプログラム生成手段によって、他プログラムモジュールが他プログラムモジュールに含まれる他プログラムモジュール用データメモリ領域をアクセスする際に、設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのア

アクセスプログラムが生成される。

【0021】

したがって、呼び出し元プログラムモジュールから他プログラムモジュールへの再入が発生したとしても、各プログラムモジュール用データメモリ領域アドレステーブルを介してアクセスするので、例えば外部変数へのアクセスで衝突が発生することは起こらず、プログラムモジュールを再入可能に構成することができる。

【0022】

また、上記のコンパイラプログラムにおいて、前記原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第1のサブプログラムから呼び出される第2のサブプログラムを含む他プログラムモジュールの記述に基づいて、処理を短縮するか否かを判別する判別手段としてコンピュータをさらに機能させ、前記判別手段によって処理を短縮しないと判別された場合、前記原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第1のサブプログラムから他プログラムモジュールに含まれる第2のサブプログラムが呼び出された後、前記アドレス設定プログラム生成手段は、前記呼び出し元プログラムモジュールの実行単位に記憶している各プログラムモジュール用データメモリ領域アドレステーブルから前記他プログラムモジュール用データメモリ領域アドレスを読み出して設定するためのアドレス設定プログラムを生成し、前記アクセスプログラム生成手段は、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスする際に、前記設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムを生成し、前記判別手段によって処理を短縮すると判別された場合、前記アクセスプログラム生成手段は、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスする際に、前記呼び出し元プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムを生成することが好ましい。

【0023】

この構成によれば、判別手段によって、原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第1のサブプログラムから呼び出される第2のサブプログラムを含む他プログラムモジュールの記述に基づいて、処理を短縮するか否かが判別される。判別手段によって処理を短縮しないと判別された場合、原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第1のサブプログラムから他プログラムモジュールに含まれる第2のサブプログラムが呼び出された後、アドレス設定プログラム生成手段によって、呼び出し元プログラムモジュールの実行単位に記憶している各プログラムモジュール用データメモリ領域アドレステーブルから他プログラムモジュール用データメモリ領域アドレスを読み出して設定するためのアドレス設定プログラムが生成される。そして、アクセスプログラム生成手段によって、他プログラムモジュールが他プログラムモジュールに含まれる他プログラムモジュール用データメモリ領域をアクセスする際に、設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムが生成される。

【0024】

一方、判別手段によって処理を短縮すると判別された場合、アクセスプログラム生成手段によって、他プログラムモジュールが他プログラムモジュールに含まれる他プログラムモジュール用データメモリ領域をアクセスする際に、呼び出し元プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムが生成される。

【0025】

したがって、呼び出し元プログラムモジュールに含まれる第1のサブプログラムから呼び出される第2のサブプログラムを含む他プログラムモジュールの記述に基づいて、処理を短縮するか否かが判別されるので、必ずしも呼び出し元プログラムモジュールと呼び出

される他プログラムモジュールとが異なるデータ領域を使用することが必要でないケースでは、各プログラムモジュール用データメモリ領域アドレステーブルを介したアドレスの設定の処理を省略し、目的プログラムの実行を高速化することができ、不要な処理を出力しないことで目的プログラムのサイズを小さくすることができる。

【0026】

また、上記のコンパイラプログラムにおいて、前記原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第1のサブプログラムから呼び出される第2のサブプログラムを含む他プログラムモジュールの記述は、前記第2のサブプログラム毎に記述される宣言であることが好ましい。

【0027】

この構成によれば、原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第1のサブプログラムから呼び出される第2のサブプログラムを含む他プログラムモジュールの記述が、第2のサブプログラム毎に記述される宣言であるので、この宣言の有無により、第2のサブプログラムの処理内容に応じてアドレスの設定の処理を省略することができ、目的プログラムの実行を高速化することができる。

【0028】

また、上記のコンパイラプログラムにおいて、前記呼び出し元プログラムモジュールによって呼び出される第2のサブプログラムを含む他プログラムモジュールを含む前記原始プログラムは、前記他プログラムモジュールに含まれる外部変数が前記呼び出し元プログラムモジュールの複数の実行によって共通にアクセスする前記他プログラムモジュール用の実行共有データメモリ領域、及び前記他プログラムモジュールに含まれる外部変数が前記呼び出し元プログラムモジュールの実行毎に固有にアクセスする前記他プログラムモジュール用の実行固有データメモリ領域のうちのいずれを使用するのかを特定するための記述を含むことが好ましい。

【0029】

この構成によれば、呼び出し元プログラムモジュールによって呼び出される第2のサブプログラムを含む他プログラムモジュールを含む原始プログラムに含まれる記述によって、他プログラムモジュールに含まれる外部変数が呼び出し元プログラムモジュールの複数の実行によって共通にアクセスする他プログラムモジュール用の実行共有データメモリ領域、及び他プログラムモジュールに含まれる外部変数が呼び出し元プログラムモジュールの実行毎に固有にアクセスする他プログラムモジュール用の実行固有データメモリ領域のうちのいずれを使用するのが特定される。

【0030】

したがって、他プログラムモジュールが複数の呼び出し元プログラムモジュールから呼び出されたとしても、各呼び出し元プログラムモジュール毎に設けられた実行固有データメモリ領域にアクセスするため、例えば外部変数へのアクセスで衝突が発生することは起こらず、プログラムモジュールを再入可能に構成することができる。また、他プログラムモジュールが複数の呼び出し元プログラムモジュールから呼び出されたとしても、各呼び出し元プログラムモジュール毎で共有する実行共有データメモリ領域にアクセスするため、例えば外部変数の幾つかを共通に利用することができる。

【0031】

また、上記のコンパイラプログラムにおいて、前記他プログラムモジュール用の実行共有データメモリ領域、及び前記他プログラムモジュール用の実行固有データメモリ領域のうちのいずれを使用するのかを特定するための記述は、前記他プログラムモジュールに含まれる外部変数毎に記述される宣言であることが好ましい。

【0032】

この構成によれば、他プログラムモジュール用の実行共有データメモリ領域、及び他プログラムモジュール用の実行固有データメモリ領域のうちのいずれを使用するのかを特定するための記述が、他プログラムモジュールに含まれる外部変数毎に記述される宣言であるので、この宣言の有無を判断することによって容易に実行共有データメモリ領域と実行

固有データメモリ領域とを使い分けることができる。

【0033】

また、上記のコンパイラプログラムにおいて、前記アドレス保存プログラム生成手段は、前記呼び出し元プログラムモジュール用の実行共有データメモリ領域アドレスを保存するとともに、前記呼び出し元プログラムモジュール用の実行固有データメモリ領域アドレスを保存するためのアドレス保存プログラムを生成し、前記アドレス設定プログラム生成手段は、前記呼び出し元プログラムモジュールによって呼び出される他プログラムモジュール用の実行共有データメモリ領域アドレスを設定するとともに、前記呼び出し元プログラムモジュールによって呼び出される他プログラムモジュール用の実行固有データメモリ領域アドレスを設定するためのアドレス設定プログラムを生成し、前記移行プログラム生成手段は、前記第1のサブプログラムから前記第2のサブプログラムへ移行するための移行プログラムを生成し、前記アドレス再設定プログラム生成手段は、前記移行された第2のサブプログラムから前記第1のサブプログラムへの復帰後に、前記保存された呼び出し元プログラムモジュール用の実行固有データメモリ領域アドレスを読み出して再設定するとともに、前記保存された呼び出し元プログラムモジュール用の実行共有データメモリ領域アドレスを読み出して再設定するためのアドレス再設定プログラムを生成し、前記アクセスプログラム生成手段は、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用の実行固有データメモリ領域をアクセスする際に、前記設定された他プログラムモジュール用の実行固有データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするとともに、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用の実行共有データメモリ領域をアクセスする際に、前記設定された他プログラムモジュール用の実行共有データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムを生成することが好ましい。

【0034】

この構成によれば、アドレス保存プログラム生成手段によって、呼び出し元プログラムモジュール用の実行共有データメモリ領域アドレスを保存するとともに、呼び出し元プログラムモジュール用の実行固有データメモリ領域アドレスを保存するためのアドレス保存プログラムが生成される。そして、アドレス設定プログラム生成手段によって、呼び出し元プログラムモジュールによって呼び出される他プログラムモジュール用の実行共有データメモリ領域アドレスを設定するとともに、呼び出し元プログラムモジュールによって呼び出される他プログラムモジュール用の実行固有データメモリ領域アドレスを設定するためのアドレス設定プログラムが生成される。移行プログラム生成手段によって、第1のサブプログラムから第2のサブプログラムへ移行するための移行プログラムが生成される。アドレス再設定プログラム生成手段によって、移行された第2のサブプログラムから第1のサブプログラムへの復帰後に、保存された呼び出し元プログラムモジュール用の実行固有データメモリ領域アドレスを読み出して再設定するとともに、保存された呼び出し元プログラムモジュール用の実行共有データメモリ領域アドレスを読み出して再設定するためのアドレス再設定プログラムが生成される。アクセスプログラム生成手段によって、他プログラムモジュールが他プログラムモジュールに含まれる他プログラムモジュール用の実行固有データメモリ領域をアクセスする際に、設定された他プログラムモジュール用の実行固有データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするとともに、他プログラムモジュールが他プログラムモジュールに含まれる他プログラムモジュール用の実行共有データメモリ領域をアクセスする際に、設定された他プログラムモジュール用の実行共有データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムが生成される。

【0035】

したがって、実行固有データメモリ領域と実行共有データメモリ領域とを使い分けることができ、他プログラムモジュールが複数の呼び出し元プログラムモジュールによって呼び出されたとしても、例えば外部変数へのアクセスで衝突が発生することは起こらず、プ

プログラムモジュールを再入可能に構成することができ、さらに、複数のプログラムモジュールは外部変数を共有することができる。

【0036】

また、上記のコンパイラプログラムにおいて、前記原始プログラムに含まれる呼び出し元プログラムモジュールから前記他プログラムモジュールが呼び出された後、前記アドレス設定プログラム生成手段は、前記呼び出し元プログラムモジュールの実行単位に記憶している各プログラムモジュール用データメモリ領域アドレステーブルから前記他プログラムモジュール用の実行固有データメモリ領域アドレスを読み出して設定するとともに、前記呼び出し元プログラムモジュールの複数の実行が共通にアクセスする前記他プログラムモジュール用の実行共有データメモリ領域アドレスを読み出して設定するためのアドレス設定プログラムを生成し、前記アクセスプログラム生成手段は、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用の実行固有データメモリ領域をアクセスする際に、前記設定された他プログラム用の実行固有データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするとともに、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用の実行共有データメモリ領域をアクセスする際に、前記設定された他プログラムモジュール用の実行共有データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムを生成することが好ましい。

【0037】

この構成によれば、原始プログラムに含まれる呼び出し元プログラムモジュールから他プログラムモジュールが呼び出された後、アドレス設定プログラム生成手段によって、呼び出し元プログラムモジュールの実行単位に記憶している各プログラムモジュール用データメモリ領域アドレステーブルから他プログラムモジュール用の実行固有データメモリ領域アドレスを読み出して設定するとともに、呼び出し元プログラムモジュールの複数の実行が共通にアクセスする他プログラムモジュール用の実行共有データメモリ領域アドレスを読み出して設定するためのアドレス設定プログラムが生成される。アクセスプログラム生成手段によって、他プログラムモジュールが他プログラムモジュールに含まれる他プログラムモジュール用の実行固有データメモリ領域をアクセスする際に、設定された他プログラム用の実行固有データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするとともに、他プログラムモジュールが他プログラムモジュールに含まれる他プログラムモジュール用の実行共有データメモリ領域をアクセスする際に、設定された他プログラムモジュール用の実行共有データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムが生成される。

【0038】

したがって、実行固有データメモリ領域と実行共有データメモリ領域とを使い分けることができ、他プログラムモジュールが複数の呼び出し元プログラムモジュールによって呼び出されたとしても、例えば外部変数へのアクセスで衝突が発生することは起こらず、プログラムモジュールを再入可能に構成することができ、さらに、複数のプログラムモジュールは外部変数を共有することができる。

【0039】

また、上記のコンパイラプログラムにおいて、前記アドレス保存プログラム生成手段は、前記第1のサブプログラムから第2のサブプログラムへ移行するために必要なデータよりも先に、前記呼び出し元プログラムモジュール用の実行共有データメモリ領域アドレス、及び実行固有データメモリ領域アドレスをスタックメモリに保存するためのアドレス保存プログラムを生成することが好ましい。

【0040】

この構成によれば、アドレス保存プログラム生成手段によって、第1のサブプログラムから第2のサブプログラムへ移行するために必要なデータよりも先に、呼び出し元プログラムモジュール用の実行共有データメモリ領域アドレス、及び実行固有データメモリ領域アドレスをスタックメモリに保存するためのアドレス保存プログラムが生成される。

【0041】

したがって、呼び出し元プログラムモジュールによって呼び出される他プログラムモジュール側が、スタックメモリに保存された実行共有データメモリ領域アドレス及び実行固有データメモリ領域アドレスの有無に関与する必要をなくすることができる。

【0042】

また、上記のコンパイラプログラムにおいて、前記原始プログラムに含まれる前記呼び出し元プログラムモジュールと、前記呼び出し元プログラムモジュールに含まれる第1のサブプログラムから呼び出される第2のサブプログラムを含む他プログラムモジュールとは、同じプログラムモジュールであることが好ましい。

【0043】

この構成によれば、原始プログラムに含まれる呼び出し元プログラムモジュールと、呼び出し元プログラムモジュールに含まれる第1のサブプログラムから呼び出される第2のサブプログラムを含む他プログラムモジュールとが、同じプログラムモジュールである。

【0044】

したがって、同じモジュール内の呼び出しを短縮することによって、モジュール内の処理を低速化させずに済むことができる。

【0045】

また、上記のコンパイラプログラムにおいて、前記呼び出し元プログラムモジュールはコード領域とデータ領域とを含み、前記他プログラムモジュールはコード領域とデータ領域とを含み、前記呼び出し元プログラムモジュールから前記他プログラムモジュールへの呼び出し命令を受けて、前記他プログラムモジュールを識別する識別番号を取得するための識別番号取得プログラムを生成する識別番号取得プログラム生成手段と、前記取得された識別番号をインデクスとして、前記呼び出し元プログラムモジュールの識別番号と、前記呼び出し元プログラムモジュールのデータ領域の先頭アドレスとが対応付けられるとともに、前記他プログラムモジュールの識別番号と、前記他プログラムモジュールのデータ領域の先頭アドレスとが対応付けられたテーブルから前記他プログラムモジュールのデータ領域の先頭アドレスを取得するための先頭アドレス取得プログラムを生成する先頭アドレス取得プログラム生成手段と、前記取得された前記他プログラムモジュールのデータ領域の先頭アドレスと、前記呼び出し元プログラムモジュールのデータ領域の先頭アドレスとを切り替えるための先頭アドレス切替プログラムを生成するための先頭アドレス切替プログラム生成手段としてコンピュータをさらに機能させることが好ましい。

【0046】

この構成によれば、呼び出し元プログラムモジュールはコード領域とデータ領域とを含み、他プログラムモジュールはコード領域とデータ領域とを含み、識別番号取得プログラム生成手段によって、呼び出し元プログラムモジュールから他プログラムモジュールへの呼び出し命令を受けて、他プログラムモジュールを識別する識別番号を取得するための識別番号取得プログラムが生成される。先頭アドレス取得プログラム生成手段によって、取得された識別番号をインデクスとして、呼び出し元プログラムモジュールの識別番号と、呼び出し元プログラムモジュールのデータ領域の先頭アドレスとが対応付けられるとともに、他プログラムモジュールの識別番号と、他プログラムモジュールのデータ領域の先頭アドレスとが対応付けられたテーブルから他プログラムモジュールのデータ領域の先頭アドレスを取得するための先頭アドレス取得プログラムが生成される。そして、先頭アドレス切替プログラム生成手段によって、取得された他プログラムモジュールのデータ領域の先頭アドレスと、呼び出し元プログラムモジュールのデータ領域の先頭アドレスとを切り替えるための先頭アドレス切替プログラムが生成される。

【0047】

したがって、アドレス変換するためのテーブルを、ページ単位ではなくモジュール単位で行うことによって、テーブルのデータ量を削減することができ、資源制約の厳しい小型の機器に適合させることができる。

【0048】

また、上記のコンパイラプログラムにおいて、前記呼び出し元プログラムモジュールはコード領域とデータ領域とを含み、前記他プログラムモジュールはコード領域とデータ領域とを含み、前記呼び出し元プログラムモジュールを識別する識別番号と、前記他プログラムモジュールを識別する識別番号とをそれぞれに付与するための識別番号付与プログラムを生成する識別番号付与プログラム生成手段と、前記付与された前記呼び出し元プログラムモジュールの識別番号と、前記呼び出し元プログラムモジュールのデータ領域の先頭アドレスとを対応付けるとともに、前記付与された前記他プログラムモジュールの識別番号と、前記他プログラムモジュールのデータ領域の先頭アドレスとを対応付けるテーブルを作成するためのテーブル作成プログラムを生成するテーブル作成プログラム生成手段と、前記呼び出し元プログラムモジュールから前記他プログラムモジュールへの呼び出し命令を受けて、前記他プログラムモジュールの識別番号を取得するための識別番号取得プログラムを生成する識別番号取得プログラム生成手段と、前記取得された識別番号をインデクスとして前記作成されたテーブルから前記他プログラムモジュールのデータ領域の先頭アドレスを取得するための先頭アドレス取得プログラムを生成する先頭アドレス取得プログラム生成手段と、前記取得された前記他プログラムモジュールのデータ領域の先頭アドレスと、前記呼び出し元プログラムモジュールのデータ領域の先頭アドレスとを切り替えるための先頭アドレス切替プログラムを生成するための先頭アドレス切替プログラム生成手段としてコンピュータをさらに機能させることが好ましい。

【0049】

この構成によれば、呼び出し元プログラムモジュールはコード領域とデータ領域とを含み、他プログラムモジュールはコード領域とデータ領域とを含み、識別番号付与プログラム生成手段によって、呼び出し元プログラムモジュールを識別する識別番号と、他プログラムモジュールを識別する識別番号とをそれぞれに付与するための識別番号付与プログラムが生成される。テーブル作成プログラム生成手段によって、付与された呼び出し元プログラムモジュールの識別番号と、呼び出し元プログラムモジュールのデータ領域の先頭アドレスとを対応付けるとともに、付与された他プログラムモジュールの識別番号と、他プログラムモジュールのデータ領域の先頭アドレスとを対応付けるテーブルを作成するためのテーブル作成プログラムが生成される。識別番号取得プログラム生成手段によって、呼び出し元プログラムモジュールから他プログラムモジュールへの呼び出し命令を受けて、他プログラムモジュールの識別番号を取得するための識別番号取得プログラムが生成される。先頭アドレス取得プログラム生成手段によって、取得された識別番号をインデクスとして、作成されたテーブルから他プログラムモジュールのデータ領域の先頭アドレスを取得するための先頭アドレス取得プログラムが生成される。先頭アドレス切替プログラム生成手段によって、取得された他プログラムモジュールのデータ領域の先頭アドレスと、呼び出し元プログラムモジュールのデータ領域の先頭アドレスとを切り替えるための先頭アドレス切替プログラムが生成される。

【0050】

したがって、アドレス変換するためのテーブルを、ページ単位ではなくモジュール単位で行うことによって、テーブルのデータ量を削減することができ、資源制約の厳しい小型の機器に適合させることができる。

【0051】

また、上記のコンパイラプログラムにおいて、前記他プログラムモジュール外で定義された変数又は関数へのアドレスを保持するためのアドレス保持プログラムを生成するアドレス保持プログラム生成手段としてコンピュータをさらに機能させ、前記アクセスプログラム生成手段は、前記他プログラムモジュール内で定義された変数へアクセスする場合、前記取得された先頭アドレスからの相対アドレスでアクセスし、前記他プログラムモジュール内で定義された関数へアクセスする場合、プログラムカウンタからの相対アドレスでアクセスし、前記他プログラムモジュール外で定義された変数又は関数へアクセスする場合、前記保持されたアドレスに前記取得された先頭アドレスからの相対アドレスでアクセスした後、前記保持されたアドレス経由で間接アクセスするためのアクセスプログラムを

生成することが好ましい。

【0052】

この構成によれば、アドレス保持プログラム生成手段によって、他プログラムモジュール外で定義された変数又は関数へのアドレスを保持するためのアドレス保持プログラムが生成される。そして、アクセスプログラム生成手段によって、他プログラムモジュール内で定義された変数又は関数へアクセスする場合、取得された先頭アドレスからの相対アドレスでアクセスされ、他プログラムモジュール内で定義された関数へアクセスする場合、プログラムカウンタからの相対アドレスでアクセスされ、他プログラムモジュール外で定義された変数又は関数へアクセスする場合、保持されたアドレスに取得された先頭アドレスからの相対アドレスでアクセスした後、保持されたアドレス経由で間接アクセスするためのアクセスプログラムが生成される。したがって、他プログラムモジュール外で定義された変数又は関数へアクセスする場合に、保持されたアドレス経由で間接的にアクセスすることができる。

【0053】

また、上記のコンパイラプログラムにおいて、少なくとも1つのモジュールで構成され、かつ現在実行中であるアプリケーションを終了するための終了プログラムを生成する終了プログラム生成手段と、アプリケーションが終了された後、メモリをコンパクションするためのコンパクションプログラムを生成するコンパクションプログラム生成手段と、メモリがコンパクションされた後、前記アプリケーションを最初から再開するための再開プログラムを生成する再開プログラム生成手段としてコンピュータをさらに機能させることが好ましい。

【0054】

この構成によれば、終了プログラム生成手段によって、少なくとも1つのモジュールで構成され、かつ現在実行中であるアプリケーションを終了するための終了プログラムが生成される。そして、アプリケーションが終了された後、コンパクションプログラム生成手段によって、メモリをコンパクションするためのコンパクションプログラムが生成され、メモリがコンパクションされた後、再開プログラム生成手段によって、アプリケーションを最初から再開するための再開プログラムが生成される。したがって、メモリのフラグメンテーションを解消することができ、メモリを有効的に利用することができる。

【0055】

また、上記のコンパイラプログラムにおいて、少なくとも1つのモジュールで構成され、かつ現在実行中であるアプリケーションを終了するための終了プログラムを生成する終了プログラム生成手段と、終了されるまでの前記アプリケーションのアドレス値に依存しない情報を保存するための保存プログラムを生成する保存プログラム生成手段と、メモリをコンパクションするためのコンパクションプログラムを生成するコンパクションプログラム生成手段と、メモリがコンパクションされた後、前記保存された情報を読み出し、読み出された情報に基づいて前記アプリケーションが終了した時点における状態まで当該アプリケーションを実行するための実行プログラムを生成する実行プログラム生成手段としてコンピュータをさらに機能させることが好ましい。

【0056】

この構成によれば、終了プログラム生成手段によって、少なくとも1つのモジュールで構成され、かつ現在実行中であるアプリケーションを終了するための終了プログラムが生成される。そして、保存プログラム生成手段によって、終了されるまでのアプリケーションのアドレス値に依存しない情報を保存するための保存プログラムが生成される。コンパクションプログラム生成手段によって、メモリをコンパクションするためのコンパクションプログラムが生成される。メモリがコンパクションされた後、実行プログラム生成手段によって、保存された情報を読み出し、読み出された情報に基づいてアプリケーションが終了した時点における状態まで当該アプリケーションを実行するための実行プログラムが生成される。したがって、メモリのフラグメンテーションを解消することができ、メモリを有効的に利用することができる。

【0057】

また、上記のコンパイラプログラムにおいて、メモリ内において、アドレス値を設定するか否かを示す識別子を設け、コンパクション前のメモリの状態を記憶するためのメモリ状態記憶プログラムを生成するメモリ状態記憶プログラム生成手段と、メモリをコンパクションするためのコンパクションプログラムを生成するコンパクションプログラム生成手段と、メモリがコンパクションされた後、前記識別子に基づいてアドレス値が設定されているエントリに対して、前記記憶されているコンパクション前のメモリの状態を参照して、アドレス値の再配置を行うための再配置プログラムを生成する再配置プログラム生成手段としてコンピュータをさらに機能させることが好ましい。

【0058】

この構成によれば、メモリ内において、アドレス値を設定するか否かを示す識別子を設け、メモリ状態記憶プログラム生成手段によって、コンパクション前のメモリの状態を記憶するためのメモリ状態記憶プログラムが生成される。そして、コンパクションプログラム生成手段によって、メモリをコンパクションするためのコンパクションプログラムが生成される。さらに、メモリがコンパクションされた後、再配置プログラム生成手段によって、識別子に基づいてアドレス値が設定されているエントリに対して、記憶されているコンパクション前のメモリの状態を参照して、アドレス値の再配置を行うための再配置プログラムが生成される。したがって、メモリのフラグメンテーションを解消することができ、メモリを有効的に利用することができる。

【0059】

また、上記のコンパイラプログラムにおいて、アドレス値を格納するアドレス値用メモリと、アドレス値以外を格納する非アドレス値用メモリとを設け、コンパクション前のアドレス値用メモリ及び非アドレス値用メモリの状態を記憶するためのメモリ状態記憶プログラムを生成するメモリ状態記憶プログラム生成手段と、前記アドレス値用メモリ及び前記非アドレス値用メモリをコンパクションするためのコンパクションプログラムを生成するコンパクションプログラム生成手段と、前記アドレス値用メモリ及び前記非アドレス値用メモリがコンパクションされた後、前記記憶されているコンパクション前のアドレス値用メモリの状態を参照して、前記アドレス値用メモリのみのアドレス値の再配置を行うための再配置プログラムを生成する再配置プログラム生成手段としてコンピュータをさらに機能させることが好ましい。

【0060】

この構成によれば、アドレス値を格納するアドレス値用メモリと、アドレス値以外を格納する非アドレス値用メモリとを設け、アドレス値用メモリ状態記憶プログラム生成手段によって、コンパクション前のアドレス値用メモリ及び非アドレス値用メモリの状態を記憶するためのアドレス値用メモリ状態記憶プログラムが生成される。コンパクションプログラム生成手段によって、アドレス値用メモリ及び非アドレス値用メモリをコンパクションするためのコンパクションプログラムが生成される。アドレス値用メモリ及び非アドレス値用メモリがコンパクションされた後、再配置プログラム生成手段によって、記憶されているコンパクション前のアドレス値用メモリの状態を参照して、アドレス値用メモリのみのアドレス値の再配置を行うための再配置プログラムが生成される。したがって、メモリのフラグメンテーションを解消することができ、メモリを有効的に利用することができる。

【0061】

また、上記のコンパイラプログラムにおいて、メモリ内において、モジュールを識別するための識別番号と、前記モジュールのデータ領域へのアドレスを設定するか、コード領域へのアドレスを設定するかを識別する識別子とを設け、ポインタにアドレス値を代入する際に前記識別番号と前記識別子とを設定するための設定プログラムを生成する設定プログラム生成手段と、前記メモリをコンパクションするためのコンパクションプログラムを生成するコンパクションプログラム生成手段と、前記メモリがコンパクションされた後、前記識別番号及び前記識別子に基づいてアドレス値を再計算するための再計算プログラム

を生成する再計算プログラム生成手段としてコンピュータをさらに機能させることが好ましい。

【0062】

この構成によれば、メモリ内において、モジュールを識別するための識別番号と、モジュールのデータ領域へのアドレスを設定するか、コード領域へのアドレスを設定するかを識別する識別子とを設け、設定プログラム生成手段によって、ポインタにアドレス値を代入する際に識別番号と識別子とを設定するための設定プログラムが生成される。コンパクションプログラム生成手段によって、メモリをコンパクションするためのコンパクションプログラムが生成される。メモリがコンパクションされた後、再計算プログラム生成手段によって、識別番号及び識別子に基づいてアドレス値を再計算するための再計算プログラムが生成される。したがって、メモリのフラグメンテーションを解消することができ、メモリを有効的に利用することができる。

【0063】

また、上記のコンパイラプログラムにおいて、メモリ内において、モジュールを識別するための識別番号と、前記モジュールのデータ領域へのアドレスを設定するか、コード領域へのアドレスを設定するかを識別する識別子とを設け、ポインタにアドレス値を代入する際に前記識別番号と前記識別子とを設定するとともに、データ領域の先頭アドレス又はコード領域の先頭アドレスからのオフセット値を設定するためのオフセット値設定プログラムを生成するオフセット値設定プログラム生成手段と、前記ポインタを使用する際に、前記オフセット値にデータ領域の先頭アドレス又はコード領域の先頭アドレスを加算することによって実アドレスに変換するための実アドレス変換プログラムを生成する実アドレス変換プログラム生成手段としてコンピュータをさらに機能させることが好ましい。

【0064】

この構成によれば、メモリ内において、モジュールを識別するための識別番号と、モジュールのデータ領域へのアドレスを設定するか、コード領域へのアドレスを設定するかを識別する識別子とを設け、オフセット値設定プログラム生成手段によって、ポインタにアドレス値を代入する際に識別番号と識別子とを設定するとともに、データ領域の先頭アドレス又はコード領域の先頭アドレスからのオフセット値を設定するためのオフセット値設定プログラムが生成される。ポインタを使用する際に、実アドレス変換プログラム生成手段によって、オフセット値にデータ領域の先頭アドレス又はコード領域の先頭アドレスを加算することによって実アドレスに変換するための実アドレス変換プログラムが生成される。したがって、メモリのフラグメンテーションを解消することができ、メモリを有効的に利用することができる。

【0065】

また、上記のコンパイラプログラムにおいて、モジュールのコード領域又はデータ領域毎にコンパクションするためのコンパクションプログラムを生成するコンパクションプログラム生成手段と、コンパクションによって移動する前記コード領域又は前記データ領域を指すハンドルがあるか否かを検索するための検索プログラムを生成する検索プログラム生成手段と、前記コード領域又は前記データ領域を指すハンドルが発見された場合、当該ハンドルを再配置するためのハンドル再配置プログラムを生成するハンドル再配置プログラム生成手段としてコンピュータをさらに機能させ、前記コンパクションプログラム生成手段は、全てのハンドルについて再配置を行った後、当該コード領域又はデータ領域をコンパクションするためのコンパクションプログラムを生成することが好ましい。

【0066】

この構成によれば、検索プログラム生成手段によって、コンパクションによって移動するコード領域又はデータ領域を指すハンドルがあるか否かを検索するための検索プログラムが生成される。コード領域又はデータ領域を指すハンドルが発見された場合、ハンドル再配置プログラム生成手段によって、当該ハンドルを再配置するためのハンドル再配置プログラムが生成される。全てのハンドルについて再配置を行った後、コンパクションプログラム生成手段によって、当該コード領域又はデータ領域をコンパクションするためのコ

ンパクションプログラムが生成される。したがって、メモリのフラグメンテーションを解消することができ、メモリを有効的に利用することができる。

【0067】

また、本発明に係るコンパイラプログラムを記録したコンピュータ読み取り可能な記録媒体は、原始プログラムを目的プログラムに変換するためのコンパイラプログラムを記録したコンピュータ読み取り可能な記録媒体であって、前記原始プログラムに含まれる呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するためのアドレス保存プログラムを生成するアドレス保存プログラム生成手段と、前記呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスを設定するためのアドレス設定プログラムを生成するアドレス設定プログラム生成手段と、前記呼び出し元プログラムモジュールに含まれる第1のサブプログラムから前記他プログラムモジュールに含まれる第2のサブプログラムへ移行するための移行プログラムを生成する移行プログラム生成手段と、前記移行された第2のサブプログラムから第1のサブプログラムへの復帰後に、前記保存された呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するためのアドレス再設定プログラムを生成するアドレス再設定プログラム生成手段と、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスする際に、前記設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムを生成するアクセスプログラム生成手段としてコンピュータを機能させるコンパイラプログラムを記録している。

【0068】

この構成によれば、アドレス保存プログラム生成手段によって、原始プログラムに含まれる呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するためのアドレス保存プログラムが生成される。そして、アドレス設定プログラム生成手段によって、呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスを設定するためのアドレス設定プログラムが生成される。さらに、移行プログラム生成手段によって、呼び出し元プログラムモジュールに含まれる第1のサブプログラムから他プログラムモジュールに含まれる第2のサブプログラムへ移行するための移行プログラムが生成される。アドレス再設定プログラム生成手段によって、移行された第2のサブプログラムから第1のサブプログラムへの復帰後に、保存された呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するためのアドレス再設定プログラムが生成される。アクセスプログラム生成手段によって、他プログラムモジュールが他プログラムモジュールに含まれる他プログラムモジュール用データメモリ領域をアクセスする際に、設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムが生成される。

【0069】

したがって、他プログラムモジュールに対して呼び出し元プログラムモジュールによる再入が発生したとしても、呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスが保存されるので、呼び出し元プログラムモジュール用データメモリ領域アドレスを退避させることができ、呼び出し元プログラムモジュールによって呼び出される他プログラムモジュール用データメモリ領域アドレスが設定されるので、再入可能な目的プログラムを生成することができ、モジュール間の動的リンクを実現することができる。

【0070】

また、本発明に係るコンパイル方法は、原始プログラムを目的プログラムに変換するためのコンパイル方法であって、コンピュータが、前記原始プログラムに含まれる呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域

アドレスを保存するためのアドレス保存プログラムを生成するアドレス保存プログラム生成ステップと、コンピュータが、前記呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスを設定するためのアドレス設定プログラムを生成するアドレス設定プログラム生成ステップと、コンピュータが、前記呼び出し元プログラムモジュールに含まれる第1のサブプログラムから前記他プログラムモジュールに含まれる第2のサブプログラムへ移行するための移行プログラムを生成する移行プログラム生成ステップと、コンピュータが、前記移行された第2のサブプログラムから第1のサブプログラムへの復帰後に、前記保存された呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するためのアドレス再設定プログラムを生成するアドレス再設定プログラム生成ステップと、コンピュータが、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスする際に、前記設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムを生成するアクセスプログラム生成ステップとを含む。

【0071】

この構成によれば、アドレス保存プログラム生成ステップにおいて、原始プログラムに含まれる呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するためのアドレス保存プログラムが生成される。そして、アドレス設定プログラム生成ステップにおいて、呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスを設定するためのアドレス設定プログラムが生成される。さらに、移行プログラム生成ステップにおいて、呼び出し元プログラムモジュールに含まれる第1のサブプログラムから他プログラムモジュールに含まれる第2のサブプログラムへ移行するための移行プログラムが生成される。アドレス再設定プログラム生成ステップにおいて、移行された第2のサブプログラムから第1のサブプログラムへの復帰後に、保存された呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するためのアドレス再設定プログラムが生成される。アクセスプログラム生成ステップにおいて、他プログラムモジュールが他プログラムモジュールに含まれる他プログラムモジュール用データメモリ領域をアクセスする際に、設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムが生成される。

【0072】

したがって、他プログラムモジュールに対して呼び出し元プログラムモジュールによる再入が発生したとしても、呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスが保存されるので、呼び出し元プログラムモジュール用データメモリ領域アドレスを退避させることができ、呼び出し元プログラムモジュールによって呼び出される他プログラムモジュール用データメモリ領域アドレスが設定されるので、再入可能な目的プログラムを生成することができ、モジュール間の動的リンクを実現することができる。

【0073】

また、本発明に係るコンパイル装置は、原始プログラムを目的プログラムに変換するコンパイル装置であって、前記原始プログラムに含まれる呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するためのアドレス保存プログラムを生成するアドレス保存プログラム生成手段と、前記呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスを設定するためのアドレス設定プログラムを生成するアドレス設定プログラム生成手段と、前記呼び出し元プログラムモジュールに含まれる第1のサブプログラムから前記他プログラムモジュールに含まれる第2のサブプログラムへ移行するための移行プログラムを生成する移行プログラム生成手段と、前記移

行された第2のサブプログラムから第1のサブプログラムへの復帰後に、前記保存された呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するためのアドレス再設定プログラムを生成するアドレス再設定プログラム生成手段と、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスする際に、前記設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムを生成するアクセスプログラム生成手段とを備える。

【0074】

この構成によれば、アドレス保存プログラム生成手段によって、原始プログラムに含まれる呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するためのアドレス保存プログラムが生成される。そして、アドレス設定プログラム生成手段によって、呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスを設定するためのアドレス設定プログラムが生成される。さらに、移行プログラム生成手段によって、呼び出し元プログラムモジュールに含まれる第1のサブプログラムから他プログラムモジュールに含まれる第2のサブプログラムへ移行するための移行プログラムが生成される。アドレス再設定プログラム生成手段によって、移行された第2のサブプログラムから第1のサブプログラムへの復帰後に、保存された呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するためのアドレス再設定プログラムが生成される。アクセスプログラム生成手段によって、他プログラムモジュールが他プログラムモジュールに含まれる他プログラムモジュール用データメモリ領域をアクセスする際に、設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムが生成される。

【0075】

したがって、他プログラムモジュールに対して呼び出し元プログラムモジュールによる再入が発生したとしても、呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスが保存されるので、呼び出し元プログラムモジュール用データメモリ領域アドレスを退避させることができ、呼び出し元プログラムモジュールによって呼び出される他プログラムモジュール用データメモリ領域アドレスが設定されるので、再入可能な目的プログラムを生成することができ、モジュール間の動的リンクを実現することができる。

【0076】

また、本発明に係る目的プログラムは、原始プログラムが変換されることによって生成される目的プログラムであって、前記原始プログラムに含まれる呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するアドレス保存手段と、前記呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスを設定するアドレス設定手段と、前記呼び出し元プログラムモジュールに含まれる第1のサブプログラムから前記他プログラムモジュールに含まれる第2のサブプログラムへ移行する移行手段と、前記移行手段によって移行された第2のサブプログラムから第1のサブプログラムへの復帰後に、前記アドレス保存手段によって保存された呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するアドレス再設定手段と、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスする際に、前記アドレス設定手段によって設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするアクセス手段としてコンピュータを機能させる。

【0077】

この構成によれば、アドレス保存手段によって、原始プログラムに含まれる呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域ア

ドレスが保存される。そして、アドレス設定手段によって、呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスが設定される。さらに、移行手段によって、呼び出し元プログラムモジュールに含まれる第1のサブプログラムから他プログラムモジュールに含まれる第2のサブプログラムへ移行される。アドレス再設定手段によって、移行された第2のサブプログラムから第1のサブプログラムへの復帰後に、保存された呼び出し元プログラムモジュール用データメモリ領域アドレスが読み出されて再設定される。アクセス手段によって、他プログラムモジュールが他プログラムモジュールに含まれる他プログラムモジュール用データメモリ領域をアクセスする際に、設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域がアクセスされる。

【0078】

したがって、他プログラムモジュールに対して呼び出し元プログラムモジュールによる再入が発生したとしても、呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスが保存されるので、呼び出し元プログラムモジュール用データメモリ領域アドレスを退避させることができ、呼び出し元プログラムモジュールによって呼び出される他プログラムモジュール用データメモリ領域アドレスが設定されるので、再入可能な目的プログラムを生成することができ、モジュール間の動的リンクを実現することができる。

【0079】

また、本発明に係る目的プログラムを記録したコンピュータ読み取り可能な記録媒体は、原始プログラムが変換されることによって生成される目的プログラムを記録したコンピュータ読み取り可能な記録媒体であって、前記原始プログラムに含まれる呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するアドレス保存手段と、前記呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスを設定するアドレス設定手段と、前記呼び出し元プログラムモジュールに含まれる第1のサブプログラムから前記他プログラムモジュールに含まれる第2のサブプログラムへ移行する移行手段と、前記移行手段によって移行された第2のサブプログラムから第1のサブプログラムへの復帰後に、前記アドレス保存手段によって保存された呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するアドレス再設定手段と、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスする際に、前記アドレス設定手段によって設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするアクセス手段としてコンピュータを機能させることを特徴とする目的プログラムを記録している。

【0080】

この構成によれば、アドレス保存手段によって、原始プログラムに含まれる呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスが保存される。そして、アドレス設定手段によって、呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスが設定される。さらに、移行手段によって、呼び出し元プログラムモジュールに含まれる第1のサブプログラムから他プログラムモジュールに含まれる第2のサブプログラムへ移行される。アドレス再設定手段によって、移行された第2のサブプログラムから第1のサブプログラムへの復帰後に、保存された呼び出し元プログラムモジュール用データメモリ領域アドレスが読み出されて再設定される。アクセス手段によって、他プログラムモジュールが他プログラムモジュールに含まれる他プログラムモジュール用データメモリ領域をアクセスする際に、設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域がアクセスされる。

【0081】

したがって、他プログラムモジュールに対して呼び出し元プログラムモジュールによる

再入が発生したとしても、呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスが保存されるので、呼び出し元プログラムモジュール用データメモリ領域アドレスを退避させることができ、呼び出し元プログラムモジュールによって呼び出される他プログラムモジュール用データメモリ領域アドレスが設定されるので、再入可能な目的プログラムを生成することができ、モジュール間の動的リンクを実現することができる。

【0082】

また、本発明に係る目的プログラム実行方法は、原始プログラムが変換されることによって生成される目的プログラムを実行する目的プログラム実行方法であって、コンピュータが、前記原始プログラムに含まれる呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するアドレス保存ステップと、コンピュータが、前記呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスを設定するアドレス設定ステップと、コンピュータが、前記呼び出し元プログラムモジュールに含まれる第1のサブプログラムから前記他プログラムモジュールに含まれる第2のサブプログラムへ移行する移行ステップと、コンピュータが、前記移行ステップにおいて移行された第2のサブプログラムから第1のサブプログラムへの復帰後に、前記アドレス保存ステップにおいて保存された呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するアドレス再設定ステップと、コンピュータが、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスする際に、前記アドレス設定ステップにおいて設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするアクセスステップとを含む。

【0083】

この構成によれば、アドレス保存ステップにおいて、原始プログラムに含まれる呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスが保存される。そして、アドレス設定ステップにおいて、呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスが設定される。さらに、移行ステップにおいて、呼び出し元プログラムモジュールに含まれる第1のサブプログラムから他プログラムモジュールに含まれる第2のサブプログラムへ移行される。アドレス再設定ステップにおいて、移行された第2のサブプログラムから第1のサブプログラムへの復帰後に、保存された呼び出し元プログラムモジュール用データメモリ領域アドレスが読み出されて再設定される。アクセスステップにおいて、他プログラムモジュールが他プログラムモジュールに含まれる他プログラムモジュール用データメモリ領域をアクセスする際に、設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域がアクセスされる。

【0084】

したがって、他プログラムモジュールに対して呼び出し元プログラムモジュールによる再入が発生したとしても、呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスが保存されるので、呼び出し元プログラムモジュール用データメモリ領域アドレスを退避させることができ、呼び出し元プログラムモジュールによって呼び出される他プログラムモジュール用データメモリ領域アドレスが設定されるので、再入可能な目的プログラムを生成することができ、モジュール間の動的リンクを実現することができる。

【0085】

また、本発明に係る目的プログラム実行装置は、原始プログラムが変換されることによって生成される目的プログラムを実行する目的プログラム実行装置であって、前記原始プログラムに含まれる呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するアドレス保存手段と、前記呼び出し元プ

プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスを設定するアドレス設定手段と、前記呼び出し元プログラムモジュールに含まれる第1のサブプログラムから前記他プログラムモジュールに含まれる第2のサブプログラムへ移行する移行手段と、前記移行手段によって移行された第2のサブプログラムから第1のサブプログラムへの復帰後に、前記アドレス保存手段によって保存された呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するアドレス再設定手段と、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスする際に、前記アドレス設定手段によって設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするアクセス手段とを備える。

【0086】

この構成によれば、アドレス保存手段によって、原始プログラムに含まれる呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスが保存される。そして、アドレス設定手段によって、呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスが設定される。さらに、移行手段によって、呼び出し元プログラムモジュールに含まれる第1のサブプログラムから他プログラムモジュールに含まれる第2のサブプログラムへ移行される。アドレス再設定手段によって、移行された第2のサブプログラムから第1のサブプログラムへの復帰後に、保存された呼び出し元プログラムモジュール用データメモリ領域アドレスが読み出されて再設定される。アクセス手段によって、他プログラムモジュールが他プログラムモジュールに含まれる他プログラムモジュール用データメモリ領域をアクセスする際に、設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域がアクセスされる。

【0087】

したがって、他プログラムモジュールに対して呼び出し元プログラムモジュールによる再入が発生したとしても、呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスが保存されるので、呼び出し元プログラムモジュール用データメモリ領域アドレスを退避させることができ、呼び出し元プログラムモジュールによって呼び出される他プログラムモジュール用データメモリ領域アドレスが設定されるので、再入可能な目的プログラムを生成することができ、モジュール間の動的リンクを実現することができる。

【0088】

また、上記の目的プログラム実行装置において、前記呼び出し元プログラムモジュールはコード領域とデータ領域とを含み、前記他プログラムモジュールはコード領域とデータ領域とを含み、前記呼び出し元プログラムモジュールを識別する識別番号と、前記他プログラムモジュールを識別する識別番号とをそれぞれに付与する識別番号付与手段と、前記識別番号付与手段によって付与された前記呼び出し元プログラムモジュールの識別番号と、前記呼び出し元プログラムモジュールのデータ領域の先頭アドレスとを対応付けるとともに、前記識別番号付与手段によって付与された前記他プログラムモジュールの識別番号と、前記他プログラムモジュールのデータ領域の先頭アドレスとを対応付けるテーブルを作成するテーブル作成手段と、前記呼び出し元プログラムモジュールから前記他プログラムモジュールへの呼び出し命令を受けて、前記他プログラムモジュールの識別番号を取得する識別番号取得手段と、前記識別番号取得手段によって取得された識別番号をインデクスとして、前記テーブル作成手段によって作成されたテーブルから前記他プログラムモジュールのデータ領域の先頭アドレスを取得する先頭アドレス取得手段と、前記先頭アドレス取得手段によって取得された前記他プログラムモジュールのデータ領域の先頭アドレスと、前記呼び出し元プログラムモジュールのデータ領域の先頭アドレスとを切り替える先頭アドレス切替手段とをさらに備えることが好ましい。

【0089】

この構成によれば、呼び出し元プログラムモジュールはコード領域とデータ領域とを含み、他プログラムモジュールはコード領域とデータ領域とを含み、識別番号付与手段によって、呼び出し元プログラムモジュールを識別する識別番号と、他プログラムモジュールを識別する識別番号とがそれぞれに付与される。そして、テーブル作成手段によって、付与された呼び出し元プログラムモジュールの識別番号と、呼び出し元プログラムモジュールのデータ領域の先頭アドレスとを対応付けるとともに、付与された他プログラムモジュールの識別番号と、他プログラムモジュールのデータ領域の先頭アドレスとを対応付けるテーブルが作成される。識別番号取得手段によって、呼び出し元プログラムモジュールから他プログラムモジュールへの呼び出し命令を受けて、他プログラムモジュールの識別番号が取得される。先頭アドレス取得手段によって、取得された識別番号をインデクスとして、作成されたテーブルから他プログラムモジュールのデータ領域の先頭アドレスが取得される。先頭アドレス切替手段によって、取得された他プログラムモジュールのデータ領域の先頭アドレスと、呼び出し元プログラムモジュールのデータ領域の先頭アドレスとが切り替えられる。

【0090】

したがって、アドレス変換するためのテーブルを、ページ単位ではなくモジュール単位で行うことによって、テーブルのデータ量を削減することができ、資源制約の厳しい小型の機器に適合させることができる。

【0091】

また、上記の目的プログラム実行装置において、前記呼び出し元プログラムモジュールはコード領域とデータ領域とを含み、前記他プログラムモジュールはコード領域とデータ領域とを含み、前記呼び出し元プログラムモジュールを識別する識別番号と、前記他プログラムモジュールを識別する識別番号とをそれぞれに付与する識別番号付与手段と、前記識別番号付与手段によって付与された前記呼び出し元プログラムモジュールの識別番号と、前記呼び出し元プログラムモジュールのデータ領域の先頭アドレスとを対応付けるとともに、前記識別番号付与手段によって付与された前記他プログラムモジュールの識別番号と、前記他プログラムモジュールのデータ領域の先頭アドレスとを対応付けるテーブルを作成するテーブル作成手段とをさらに備えることが好ましい。

【0092】

この構成によれば、呼び出し元プログラムモジュールはコード領域とデータ領域とを含み、他プログラムモジュールはコード領域とデータ領域とを含み、識別番号付与手段によって、呼び出し元プログラムモジュールを識別する識別番号と、他プログラムモジュールを識別する識別番号とがそれぞれに付与される。そして、テーブル作成手段によって、付与された呼び出し元プログラムモジュールの識別番号と、呼び出し元プログラムモジュールのデータ領域の先頭アドレスとを対応付けるとともに、付与された他プログラムモジュールの識別番号と、他プログラムモジュールのデータ領域の先頭アドレスとを対応付けるテーブルが作成される。

【0093】

したがって、アドレス変換するためのテーブルを、ページ単位ではなくモジュール単位で行うことによって、テーブルのデータ量を削減することができ、資源制約の厳しい小型の機器に適合させることができる。

【発明の効果】

【0094】

本発明によれば、いわゆる機器組み込みプログラムのようにオブジェクトプログラムを格納することのできるメモリ領域が限られており、しかも近年の携帯情報機器のようにその限られたメモリ領域に高度且つ高機能な情報処理のためのオブジェクトプログラムを格納しなければならないような目的に適う、再入可能なオブジェクトプログラムを自動生成することができ、モジュール間の動的なリンクを実現することができる。

【発明を実施するための最良の形態】

【0095】

本特許出願に係る発明（以後、「本発明」とも言う）の実施の形態について図を参照して説明する。

【0096】

（第1の実施の形態）

図1は、本発明の一実施の形態による動的モジュールリンクシステムの構成を示す図である。図1に示す動的モジュールリンクシステム10は、サーバ11と複数の端末装置12（12a, 12b, 12c, ...）とを備えて構成される。サーバ11と各端末装置12とは、例えば、インターネット13を介して接続されている。なお、本実施の形態における端末装置とは、原始プログラムが変換されることによって生成される目的プログラムを実行する目的プログラム実行装置のことである。目的プログラム実行装置は、携帯電話機等の携帯通信端末や、インターネット13等のネットワークに接続された家電機器等を含む。

【0097】

なお、本実施の形態における端末装置は、インターネット13等のネットワークを介してサーバ11から目的プログラムを受信し、受信した目的プログラムを実行するものであるが、本発明は特にこれに限定されず、コンパイル装置によって生成された目的プログラムを、例えばCD-ROMやDVD-ROM等のコンピュータ読み取り可能な記録媒体に記録させ、端末装置が備える、例えばCD-ROMドライブやDVD-ROMドライブ等の記録媒体駆動装置によって、前期記録媒体に記録された目的プログラムを読み込み、読み込まれた目的プログラムを実行してもよい。

【0098】

サーバ11は、コンパイル装置14を備えて構成される。コンパイル装置14は、プログラムによって作成された原始プログラムを目的プログラムに変換する。端末装置12は、コンパイル装置14によって変換された目的プログラムを実行する。

【0099】

図2は、図1に示すコンパイル装置14の構成を示すブロック図である。図2に示すように、コンパイル装置14は、プログラム実行部16及びプログラム記憶部17を備えて構成される。

【0100】

プログラム実行部16は、例えばCPU（中央演算処理装置）等で構成され、アドレス保存プログラム生成部16a、アドレス設定プログラム生成部16b、移行プログラム生成部16c、アドレス再設定プログラム生成部16d及びアクセスプログラム生成部16eを備えて構成される。プログラム記憶部17は、コンパイラプログラム記憶部17a、ソースプログラム記憶部17b及びオブジェクトプログラム記憶部17cを備えて構成される。

【0101】

アドレス保存プログラム生成部16aは、ソースプログラムに含まれる呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するためのアドレス保存プログラムを生成する。

【0102】

アドレス設定プログラム生成部16bは、呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスを設定するためのアドレス設定プログラムを生成する。

【0103】

移行プログラム生成部16cは、呼び出し元プログラムモジュールに含まれる第1のサブプログラムから他プログラムモジュールに含まれる第2のサブプログラムへ移行するための移行プログラムを生成する。

【0104】

アドレス再設定プログラム生成部16dは、移行された第2のサブプログラムから第1のサブプログラムへの復帰後に、保存された呼び出し元プログラムモジュール用データメ

メモリ領域アドレスを読み出して再設定するためのアドレス再設定プログラムを生成する。

【0105】

アクセスプログラム生成部16eは、他プログラムモジュールが他プログラムモジュールに含まれる他プログラムモジュール用データメモリ領域をアクセスする際に、設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムを生成する。

【0106】

コンパイラプログラム記憶部17aは、予め作成されたソースプログラム（原始プログラム）をオブジェクトプログラム（目的プログラム）に変換するためのコンパイラプログラムを記憶する。ソースプログラム記憶部17bは、例えば、C言語やC++言語等で記述されたソースプログラムを記憶する。なお、ソースプログラムは、アプリケーション全体ではなく、モジュール単位で作成される。オブジェクトプログラム記憶部17cは、ソースプログラムから変換されたオブジェクトプログラムを記憶する。

【0107】

次に、図2に示すコンパイル装置によるコンパイラ処理について説明する。図3は、図2に示すコンパイル装置によるコンパイラ処理の一例を示すフローチャートである。

【0108】

ステップS1において、アドレス保存プログラム生成部16aは、アプリケーションプログラムモジュールのデータ領域の先頭アドレスをスタックに保存するためのアドレス保存プログラムを生成する。

【0109】

ステップS2において、アドレス設定プログラム生成部16bは、ライブラリプログラムモジュールのデータ領域の先頭アドレスを設定するためのアドレス設定プログラムを生成する。

【0110】

ステップS3において、移行プログラム生成部16cは、アプリケーションプログラムモジュールからライブラリプログラムモジュールに移行するための移行プログラムを生成する。

【0111】

ステップS4において、アドレス再設定プログラム生成部16dは、スタックに保存したアプリケーションプログラムモジュールのデータ領域の先頭アドレスに再設定するためのアドレス再設定プログラムを生成する。

【0112】

ステップS5において、アクセスプログラム生成部16eは、ライブラリプログラムモジュールがライブラリプログラムモジュールに含まれるライブラリプログラムモジュール用データ領域をアクセスする際に、設定されたライブラリプログラムモジュール用データ領域の先頭アドレスからの相対アドレスで、データ領域をアクセスするためのアクセスプログラムを生成する。

【0113】

このように、コンパイラプログラムが実行されることによって生成されるアドレス保存プログラム、アドレス設定プログラム、移行プログラム、アドレス再設定プログラム及びアクセスプログラムがオブジェクトプログラムとして各端末装置12に送信され、端末装置12によって、受信したオブジェクトプログラムが実行される。

【0114】

図4は、本発明の第1の実施の形態におけるコンパイル装置が変換するソースプログラムと、コンパイル装置によって変換されたオブジェクトプログラムとの構造を示す図であり、図4(a)は、アプリケーションプログラムモジュールの構造を示す図であり、図4(b)は、ライブラリプログラムモジュールの構造を示す図である。

【0115】

但し、これらのプログラムは必ずしも1つにまとまった形で存在するとは限らず、幾つ

かの物理的な媒体、ファイル等に分散して記録されていることもある。

【0116】

これらのプログラムは、アプリケーションプログラムモジュール121とアプリケーションプログラムモジュール131とライブラリプログラムモジュール151と、その他本発明の説明に直接関係しないので図示していない幾つかのプログラムモジュールとを含んでいる。

【0117】

プログラムモジュールとは、プログラムを何らかの単位（例えば、機能単位、リンク単位、バージョンアップ単位等）で分割したものであり、プログラムコード領域と固有データ領域とを保持する。尚ここでは固有データ領域としたが、必ずしも厳密に固有であることは必要でなく、他の何らかの目的や用途と兼用しているものであっても良い。このことは他の説明についても同様である。

【0118】

プログラムモジュールのプログラムコード領域とは、そのプログラムモジュール内に記述されたプログラムの実行命令等を保持する領域である。

【0119】

プログラムモジュールの固有データ領域とは、そのプログラムモジュール内で定義された、外部変数、および静的変数の実体を保持する領域である。以後、外部変数と静的変数の両方を含めて、外部変数と言うこともある。

【0120】

アプリケーションプログラムモジュール121は、プログラムコード領域123とこのアプリケーションプログラムモジュール121の固有データ領域125と、その他本発明の説明に直接関係しないので図示していない幾つかの部分とを含んでいる。

【0121】

アプリケーションプログラムモジュール131は、プログラムコード領域133とこのアプリケーションプログラムモジュール131の固有データ領域135と、その他本発明の説明に直接関係しないので図示していない幾つかの部分とを含んでいる。

【0122】

これらのアプリケーションプログラムモジュール121とアプリケーションプログラムモジュール131とは、呼び出し元プログラムであって、共通にライブラリプログラムモジュール151を呼び出す。この呼び出しは通常、関数コール、サブルーチンコール等の形態で行われる。それが図4に示すプログラム呼び出し1とプログラム呼び出し2（再入：reentry）である。

【0123】

これらプログラムモジュールは通常、幾つかのサブプログラムを含んでいる。このサブプログラムは通常、関数やサブルーチンやその他である。また、これら関数やサブルーチン1つをプログラムモジュールと呼ぶこともある。

【0124】

また通常、プログラムモジュールの呼び出しは、他プログラムモジュールのサブプログラムを呼び出すことによって行うが、同一プログラムモジュール内のサブプログラムを呼び出すこともある。

【0125】

いわゆるマルチタスク環境、その他時分割的に複数のプロセスまたはタスクまたはアプリケーションが並行して実行される、或いは実行しなければならない環境では、例えば、アプリケーションプログラムモジュール121がライブラリプログラムモジュール151を呼び出して実行中に、並行して、アプリケーションプログラムモジュール131がライブラリプログラムモジュール151を呼び出して実行することが発生する。

【0126】

このような状態は種々の情報機器で無数に近く発生するが、非常に簡単な幾つかの例を示す。

【0127】

例えば、携帯電話等のモバイル情報通信機能を有する機器では、ユーザーが送信するためのデータを入力している時にも、同時に並行して着信の有無を監視していなければならない。

【0128】

このデータ入力と着信との監視を別のCPUで行うこともできるが、CPUの節約のため1つのCPUで行おうとするならば、時分割的に並行して行う必要が起こる。

【0129】

そしてある基本的な関数型サブルーチンが、これらデータ入力アプリケーションと着信監視アプリケーションとの両方で使用される関数型サブルーチンであったならば、この関数型サブルーチンは、一方のアプリケーションから呼び出されて実行中に、もう一方のアプリケーションから呼び出されることが起こり得る。

【0130】

このように複数のタスク等が時間的に重複して同一のプログラムモジュール（関数またはサブプログラムまたはサブルーチン等）を呼び出し、実行するようなケースを「再入：reentry」と言い、あるプログラムモジュールが時間的に重複・並行して複数のタスクを実行中のプログラムから呼び出され、実行され得るように構成されていることを「再入可能：reentrant」と言う。

【0131】

従来から、プログラムモジュールを再入可能であるように構成する方法は幾つか知られている。

【0132】

その第1の方法は、いわゆる外部変数を使用せず、全て内部変数（以後、Local変数と言うこともあるが同一内容を表す）で処理することであった。内部変数はこのプログラムモジュールが呼び出される毎にスタック領域に新たに確保され、このプログラムモジュールの処理が終了するのと同時に解放されるため、全て内部変数を使用している限り、変数へのアクセスが衝突することは発生せず、同時に並行的に実行されても問題はなかった。

【0133】

その第2の方法は、いわゆる外部変数を使用するが、この外部変数に対するアクセスをいわゆるセマフォ（semaphore：腕木信号機）と呼ばれる状態変数によって調停する方法である。

【0134】

第1の方法では、外部変数を使用することができないため、このプログラム以外のプログラムモジュールとデータとを受け渡しするのに制約を受けることがある。

【0135】

第2の方法では、煩雑なセマフォに対するテスト・アンド・セットを外部変数にアクセスする毎に行わなければならない、多くの負担をプログラマに課し、間違いや誤動作の原因となることが多かった。

【0136】

ライブラリプログラムモジュール151は、プログラムコード領域153とこのライブラリプログラムモジュール151の固有データ領域155、157と、その他本発明の説明に直接関係しないので図示していない幾つかの部分とを含んでいる。

【0137】

アプリケーションプログラムモジュール121用データ領域155とアプリケーションプログラムモジュール131用データ領域157とは、このライブラリプログラムモジュール151を呼び出した元のプログラムモジュールによって区別されている。

【0138】

例えば、アプリケーションプログラムモジュール121からこのライブラリプログラムモジュール151が呼び出された時には、プログラムコード領域153は外部変数領域と

してアプリケーションプログラムモジュール121用データ領域155をアクセスしてその処理を行い、アプリケーションプログラムモジュール131からこのライブラリプログラムモジュール151が呼び出された時には、プログラムコード領域153は外部変数領域としてアプリケーションプログラムモジュール131用データ領域157をアクセスしてその処理を行う。

【0139】

このように呼び出し元プログラム単位に、ライブラリプログラムモジュールが使用する固有データ領域、特に外部変数のために使用するデータ領域を区分して使用することによって、外部変数に対するアクセスが衝突することを気にせず、プログラムを作成することができる。

【0140】

なお、既に説明したように、内部変数については、このライブラリプログラムモジュール151が呼び出される毎に、領域がスタックに確保されるため、アクセスが衝突する問題は生じない。

【0141】

尚ここで、データ領域を呼び出し元アプリケーションプログラム単位に用意して使用すると説明したが、ここで言うアプリケーションプログラムを、アプリケーションタスク単位と読み替えても良く、ここで言うアプリケーションプログラムはアプリケーションタスクと同じ概念であると考えても良い。

【0142】

また、ここで言うアプリケーションプログラムを、アプリケーションプロセス単位と読み替えても良く、ここで言うアプリケーションプログラムはアプリケーションプロセスと同じ概念であると考えても良い。

【0143】

より詳しくアプリケーションプログラムモジュール121がライブラリプログラムモジュール151を呼び出す時の処理手順について図5～図8を用いて説明する。

【0144】

図5は、図1に示す端末装置12の構成を示すブロック図である。図6は、図5に示す端末装置12によってアプリケーションプログラムモジュール121がライブラリプログラムモジュール151を呼び出す処理を示すフローチャートである。図7は、アプリケーションプログラムモジュールからライブラリプログラムモジュールを呼び出す処理を説明するための図であり、図7(a)は、アプリケーションプログラムモジュールの処理を説明するための図であり、図7(b)は、ライブラリプログラムモジュールの処理を説明するための図である。図8は、アプリケーションプログラムモジュール及びライブラリプログラムモジュールのソースプログラム及びオブジェクトプログラムの一例を示す図であり、図8(a)は、アプリケーションプログラムモジュールのソースプログラム及びオブジェクトプログラムの一例を示す図であり、図8(b)は、ライブラリプログラムモジュールのソースプログラム及びオブジェクトプログラムの一例を示す図である。なお、図8(a)、(b)に示すソースプログラムは、C言語で記述されたプログラムを例として記述している。

【0145】

図5に示すように、端末装置12は、プログラム実行部18及びメモリ19を備えて構成される。プログラム実行部18は、例えばDSP (Digital Signal Processor) で構成され、アドレス保存部18a、アドレス設定部18b、移行部18c、アドレス再設定部18d及びアクセス部18eを備えて構成される。メモリ19は、アプリケーションプログラムモジュール記憶部19a、ライブラリプログラムモジュール記憶部19b及びスタック19cを備えて構成される。

【0146】

アプリケーションプログラムモジュール記憶部19aは、コンパイル装置14から送信されたアプリケーションプログラムモジュールを記憶する。ライブラリプログラムモジュ

ール記憶部19bは、コンパイル装置14から送信されたライブラリプログラムモジュールを記憶する。スタック19cは、サブルーチンや関数を呼び出す際の処理中のデータやアドレス等を一時的に記憶する。

【0147】

アドレス保存部18aは、呼び出し元プログラムモジュールであるアプリケーションプログラムモジュール121が使用するアプリケーションプログラムモジュール用固有データ領域125の先頭アドレスをスタック19cに保存する。

【0148】

アドレス設定部18bは、アプリケーションプログラムモジュール121によって呼び出されるライブラリプログラムモジュール151のアプリケーションプログラムモジュール用データ領域155の先頭アドレスを設定する。

【0149】

移行部18cは、アプリケーションプログラムモジュール121に含まれるプログラムコード領域123からライブラリプログラムモジュール151に含まれるプログラムコード領域153へ移行する。

【0150】

アドレス再設定部18dは、移行されたライブラリプログラムモジュール151のプログラムコード領域153からアプリケーションプログラムモジュール121のプログラムコード領域123への復帰後に、保存されたアプリケーションプログラムモジュール用データ領域125の先頭アドレスを読み出して再設定する。

【0151】

アクセス部18eは、ライブラリプログラムモジュール151のプログラムコード領域153がライブラリプログラムモジュール151に含まれるライブラリプログラムモジュール用データ領域155をアクセスする際に、設定されたライブラリプログラムモジュール用データ領域155の先頭アドレスからの相対アドレスで、データ領域155をアクセスする。

【0152】

ここで、アプリケーションプログラムモジュール121のソースプログラムに記述された「bar(A, B);」命令がオブジェクトプログラムに変換されると、ライブラリプログラムモジュール151に含まれる関数型サブルーチンであるbar(以後、関数型サブプログラムbarと言うこともあり、単に関数barと言うこともあるが同一である)を呼び出すオブジェクトプログラムが生成される(図7(a)の1~7)。

【0153】

図6のステップS11において、アドレス保存部18aは、現在のDP(データポイント)レジスタ(以後単にDPと言うこともある)の内容をスタック19cに保存する。アプリケーションプログラムモジュール121のプログラムコード領域123に含まれるプログラムは、ライブラリプログラムモジュール151に含まれる関数barを呼び出す際に、まず現在実行中のアプリケーションプログラムモジュール121が使用している固有データ領域125の先頭アドレスである0x0600(16進数表記で0600であり、10進数表記に変換すると、 $0 \times 16^3 + 6 \times 16^2 + 0 \times 16 + 0$ である、以後同じ)を保存しているDPレジスタの内容をスタックに保存する(図7(a)の1、図8(a)の1)。

【0154】

これはライブラリプログラムモジュール151から戻ってきた時に、アプリケーションプログラムモジュール121がDPの内容を再度0x0600に設定し、アプリケーションプログラムモジュール121が自分自身のための固有データ領域125を再度使用することができるようにするためである。尚ここでは固有のデータ領域としたが、必ずしも厳密に固有であることは必ずしも必要でなく、他の何らかの用途や目的、その他と共有であっても良い。このことは他の説明についても同様である。

【0155】

DPレジスタは、実行中のプログラムモジュールのデータ領域の先頭アドレスを保持する領域である。この領域として、通常特別なレジスタを用意するが、必ずしも特別なレジスタを用意せず、汎用的なレジスタを使用してもよい。

【0156】

次に、ステップS12において、アドレス保存部18aは、関数barの引数をスタック19cに保存する。すなわち、アプリケーションプログラムモジュール121に含まれるプログラムは、ライブラリプログラムモジュール151に引き渡すデータである引数Aと引数Bとをスタック19cに保存する（図7（a）の2、図8（a）の2）。

【0157】

ここで、この時のスタック19cの状態について説明する。図9は、スタックの状態を説明するための図であり、図9（a）は、引数Aと引数Bとをスタックに保存した際のスタックの状態を示す概念図であり、図9（b）は、スタックの使われ方を説明するための概念図である。

【0158】

図9（a）に示すようにスタック（以後、スタックメモリと言うこともある）19cはLast-in/First-out型のメモリ領域で、あるプログラムモジュールから他プログラムモジュール（例えば、関数やサブルーチン、サブプログラム、その他である）を呼び出す時に必要なデータを保存して、プログラムモジュール間でデータの引き渡しに使ったり、Local変数用メモリ領域として使用したり、呼び出された関数等から呼び出し元のプログラムに戻るためのアドレスの保存や、その他の目的に使用されるものである。

【0159】

スタック19cのデータを出し入れする領域のアドレスは、スタックポインタと呼ばれるレジスタで示される（以後、単にスタックポインタ或いはスタックポインタレジスタと呼ぶこともある）。

【0160】

このスタックの使われ方を図9（b）を用いて説明する。例えば、図9（b）に示すスタックの上から3番目に格納されているローカル変数2に対してアクセスするには、スタックポインタの値を変更するか或いは、スタックポインタレジスタの値マイナス2の値をアドレスとして間接参照を行うこともできる。

【0161】

次に、ステップS13において、アドレス設定部18bは、新しいDPを設定する。すなわち、アプリケーションプログラムモジュール121に含まれるプログラムは、自分自身の固有データ領域125内の領域に記憶している、関数barを含むライブラリプログラムモジュール151で使用する、アプリケーションプログラムモジュール121用データ領域155のアドレス（0x800）を読み出し、そのアドレスをDPに設定する（図7（a）の3、図8（a）の3）。

【0162】

前記関数barを含むライブラリプログラムモジュール151で使用する、アプリケーションプログラムモジュール121用データ領域155のアドレス（0x800）を保持する領域は、通常コンパイラが設定する。しかし必ずしも、前記領域をコンパイラが設定せず、ライブラリとして用意し、リンク時に設定してもよい。

【0163】

次に、ステップS14において、移行部18cは、ライブラリ関数を呼び出す。アプリケーションプログラムモジュール121に含まれるプログラムは、呼び出し先プログラムである関数barから自分自身に帰ってくる時のアドレス、即ち関数barへのジャンプ命令の次の命令のアドレスをスタック19cに保存し、関数barへジャンプする（図7（a）の4、図8（a）の4）。

【0164】

図7（b）に示すように、呼び出されるプログラムである関数barは、ライブラリ

プログラムモジュール151に含まれるプログラムコード領域153に含まれている。この関数barは引数としてスタック19cに保存された引数Aと引数Bとを取り出して処理に使用する。この時のスタック19cの状態は前記図9(a)に示す通りである。

【0165】

更に、ステップS15において、アクセス部18eは、外部変数に値を代入する。関数barは外部変数として、例えばこのライブラリプログラムモジュール151で定義された外部変数Cを使用する。関数barではこの外部変数Cに対してアクセスし、例えばそのメモリ領域に値10を代入するに際し、DPに保存されている値(0x800)にDPからの相対アドレス(0x50)を加算したアドレス(0x850)に値10を代入する。この命令形式を図7(b)の5と図8(b)の5に示す。

【0166】

これによって関数barは、アドレス0x800から始まるメモリ領域であるアプリケーションプログラムモジュール121用データ領域155に対してアクセスすることができる。

【0167】

関数barでの処理が終ると、プログラムの制御は元のアプリケーションプログラムモジュール121に含まれるプログラムに戻される。

【0168】

そして、ステップS16において、アドレス再設定部18dは、関数barの引数をスタック19cから破棄する。関数barからの復帰直後に、スタック19cから引数A、Bが破棄される(図7(a)の6、図8(a)の6)。尚ここでは関数barからの復帰直後としたが、必ずしも厳密に直後であることは必要でなく、何らかの処理の後であっても良い。このことは他の説明についても同様である。

【0169】

次に、ステップS17において、アドレス再設定部18dは、スタック19cに保存していた元のDPの値、この例では0x600がスタック19cから取り出されてDPに戻される(図7(a)の7、図8(a)の7)。

【0170】

これによって呼び出し元のアプリケーションプログラムモジュール121に含まれるプログラムは、再度自分自身の固有データ領域125に対してアクセスすることが可能になる。

【0171】

尚この時、図9(a), (b)に示す通り、呼び出し元のDPの値は、スタックの一番下に保存されているので、呼び出されたライブラリプログラムモジュール151の側ではその内容を読み出す必要がないと共に、実際に読み出さなくても他の必要な処理を全て行うことが可能であり、不要なアクセスを行う必要が生じない。

【0172】

また、例えば次の第2の実施の形態に示すような他の実施の形態であって、DPをスタックに保存するか保存しないか選択ができる実施の形態では、DPをスタックの一番下に保存することによって、DPがスタックに保存されているか保存されていないかに係わらず、同一アドレスに対するアクセスによって同じ処理を行うことができ、DPがスタックに保存されているか否かを判断して処理を分ける必要がない。

【0173】

DPをスタックの一番下に保存することによって、DPがスタックに保存されているか保存されていないかに係わらず、呼び出されたライブラリプログラムモジュール151の側で必要なデータ、例えば引数や帰りアドレスを取り出すスタック位置は変わらない。すなわち、引数や帰りアドレスを取り出すスタックのスタックポインタからの相対位置は変化しないので、それを考慮してこれら引数や帰りアドレスを取り出すことは必要でない。

【0174】

尚、この例では、アプリケーションプログラムモジュールから、ライブラリプログラム

モジュールを呼び出す例について書いたが、ライブラリプログラムモジュールから、他のライブラリモジュールを呼び出すことも可能であり、その際の処理は、アプリケーションプログラムモジュールから、他ライブラリモジュールを呼び出す場合の処理と同じであるから、説明を省略する。

【0175】

尚、この例では、他プログラムモジュールを呼び出す例について書いたが、同一プログラムモジュールを呼び出すことも可能であり、その際の処理は、他プログラムモジュールを呼び出す場合の処理と同じであるから、説明を省略する。

【0176】

このように、他プログラムモジュールに対して呼び出し元プログラムモジュールによる再入が発生したとしても、アプリケーションプログラムモジュール121が使用する固有データ領域125の先頭アドレスがスタックに保存されるので、アプリケーションプログラムモジュール121が使用する固有データ領域125の先頭アドレスを退避させることができ、アプリケーションプログラムモジュール121によって呼び出されるライブラリプログラムモジュールが使用するアプリケーションプログラムモジュール121用データ領域155の先頭アドレスが設定されるので、再入可能なオブジェクトプログラムを生成することができ、モジュール間の動的リンクを実現することができる。

【0177】

(第2の実施の形態)

前記本発明の第1の実施の形態では、アプリケーションプログラムモジュール121のソースプログラムに記述された「bar (A, B);」命令がオブジェクトプログラムに変換されることによって生成されるオブジェクトプログラムは、現在実行中のアプリケーションプログラムモジュール121が使用している固有データ領域125の先頭アドレスを保存しているDPレジスタの内容をスタックに保存し、ライブラリプログラムモジュール151に引き渡すデータである引数Aと引数Bをスタックに保存し、呼び出される関数barで使用する、アプリケーションプログラムモジュール121用データ領域155のアドレスを読み出してそのアドレスをDPに設定し、関数barから自分自身に帰ってくる時のアドレスをスタックに保存し、関数barへジャンプし、関数barからの復帰時には、引数Aと引数Bをスタックから破棄し、スタックに保存している元のDPの値を再設定するものであった。

【0178】

次に説明する本発明の第2の実施の形態では、上記のような処理を行うか、それとも上記の処理の一部を行わずに省略するかの記述を、ライブラリプログラムモジュール151のソースプログラムが含むものである。

【0179】

なお、上述の第1の実施の形態では、他プログラムモジュールの呼び出しを例に挙げ説明したが、本発明の第2の実施の形態では、同一モジュール内の関数barの呼び出しを例に挙げて説明する。

【0180】

図10は、第2の実施の形態におけるソースプログラム及びオブジェクトプログラムの一例を示す図であり、図10(a)は、ライブラリプログラムモジュールのソースプログラムの一例を示す図であり、図10(b)は、ライブラリプログラムモジュールのオブジェクトプログラムの一例を示す図である。なお、図10ではC言語を例に記述している。この例ではC言語の言語仕様を拡張して、新規キーワードとして「private」を追加し、ソースコード上で「private」修飾詞をつけることにより、宣言を記述している。

【0181】

例えば、図10に示す関数barの呼び出しでは、上記本発明の第1の実施の形態で行った処理の一部を行うことなく、省略できることを示す。図10(a)に示す関数barでは、処理の一部を省略するための宣言が、例えば、「private void ba

r (int A, int B)」で記述されている。

【0182】

図10(a)に示すように、ライブラリモジュールプログラムのソースプログラムに含まれる関数barには、関数barが「private」である旨の記述が含まれている。

【0183】

この記述は「関数barがいわゆるプライベート(private)関数である」旨を宣言する記述であり、例えば「プライベート(private)宣言」或いは「プライベート(private)関数宣言」と呼ばれることもある。

【0184】

このプライベート関数barを呼び出す記述がある場合、第1の実施の形態で説明したような、呼び出し元であるプログラムモジュールが使用しているDPの保存や、呼び出されるプログラムモジュールで使用される固有データ領域アドレスを示すDPの設定や、関数barからの復帰時に、スタックに保存している元のDPの値の再設定を行うためのオブジェクトプログラムは生成されない。

【0185】

図11は、本発明の第2の実施の形態におけるコンパイル装置の構成を示すブロック図である。図11に示すコンパイル装置14は、プログラム実行部16及びプログラム記憶部17を備えて構成される。

【0186】

プログラム実行部16は、例えばCPU(中央演算処理装置)等で構成され、アドレス保存プログラム生成部16a、アドレス設定プログラム生成部16b、移行プログラム生成部16c、アドレス再設定プログラム生成部16d、アクセスプログラム生成部16e及び宣言判別部16fを備えて構成される。プログラム記憶部17は、コンパイラプログラム記憶部17a、ソースプログラム記憶部17b及びオブジェクトプログラム記憶部17cを備えて構成される。

【0187】

なお、図11に示す第2の実施の形態におけるコンパイル装置と、図2に示す第1の実施の形態におけるコンパイル装置とで同じ構成物については、同じ符号を付し、以下の説明では第1の実施の形態とは異なる構成についてのみ説明する。

【0188】

宣言判別部16fは、ソースプログラムに含まれるアプリケーションプログラムモジュールに含まれるプログラムコード領域から呼び出されるライブラリプログラムモジュールのプログラムコード領域の記述に基づいて、処理を短縮するか否かを判別する。すなわち、宣言判別部16fは、ソースプログラムの記述に上述のプライベート宣言が含まれているか否かを判別する。

【0189】

ここで、宣言判別部16fによって処理を短縮しないと判別された場合、すなわち、プライベート宣言が含まれていないと判別された場合、第1の実施の形態と同様に、アドレス保存プログラム生成部16a、アドレス設定プログラム生成部16b、移行プログラム生成部16c、アドレス再設定プログラム生成部16d及びアクセスプログラム生成部16eによる処理が行われる。

【0190】

また、宣言判別部16fによって処理を短縮しないと判別された場合、すなわち、プライベート宣言が含まれていると判別された場合、移行プログラム生成部16cは、アプリケーションプログラムモジュールに含まれるプログラムコード領域からライブラリプログラムモジュールに含まれるプログラムコード領域へ移行するための移行プログラムを生成し、アクセスプログラム生成部16eは、ライブラリプログラムモジュールに含まれるプログラムコード領域がライブラリプログラムモジュールに含まれるライブラリプログラムモジュール用データ領域をアクセスする際に、ライブラリプログラムモジュール用データ

領域の先頭アドレスからの相対アドレスで、データ領域をアクセスするためのアクセスプログラムを生成する。

【0191】

図12は、図11に示すコンパイル装置によるコンパイラ処理の一例を示すフローチャートである。

【0192】

ステップS21において、宣言判別部16fは、ライブラリプログラムモジュールのプログラムコード領域の記述にプライベート宣言が含まれているか否かを判断する。ここで、プライベート宣言が含まれていないと判断されると（ステップS21でNO）、ステップS22に移行し、プライベート宣言が含まれていると判断されると（ステップS21でYES）、ステップS27に移行する。

【0193】

ステップS22からステップS26までの処理は、図3に示すステップS1からステップS5までの処理と同じであるので説明を省略する。また、ステップS27における処理も、図3に示すステップS3における処理と同じであるので説明を省略する。

【0194】

ステップS28において、アクセスプログラム生成部16eは、ライブラリプログラムモジュールに含まれるプログラムコード領域がライブラリプログラムモジュールに含まれるライブラリプログラムモジュール用データ領域をアクセスする際に、ライブラリプログラムモジュール用データ領域アドレスからの相対アドレスで、データ領域をアクセスするためのアクセスプログラムを生成する。

【0195】

図13は、本発明の第2の実施の形態におけるライブラリプログラムモジュールのコード領域及びデータ領域を示す図である。このプライベート関数barを呼び出す記述がライブラリプログラムモジュール151のプログラムコード領域153で行われると、図11に示すように、関数barで使用するためのデータである引数Aや引数Bを関数barに引き渡すために、当該引数A及び引数Bをスタックへ保存するステップと、関数barへジャンプする命令の次の命令のアドレス（戻り番地）をスタックへ格納し、関数barへジャンプするステップと、関数barからの復帰時に、引数Aや引数Bをスタックから破棄するステップとが実行されるオブジェクトプログラムが本実施の形態のコンパイル装置によって生成される。

【0196】

これによって、必ずしも呼び出し元であるプログラムモジュールと呼び出されるプログラムモジュールとが異なるデータ領域を使用することが必要でないケースでは、関数barをプライベート宣言することによって、DPの保存や設定や再設定のステップを省略し、オブジェクトプログラムの実行を高速化することができ、不要なステップを出力しないことでオブジェクトプログラムサイズを小さくすることもできる。

【0197】

このようなプライベート宣言された関数barが実行されると、図11に示すように、DPの内容は呼び出し元プログラムであるプログラムモジュールの状態のまま変化していないので、例えば外部変数Cに対してデータの読み出し等が実行されると、呼び出し元であるプログラムモジュールの固有データ領域（この場合、固有データ領域155）がアクセスされる。

【0198】

また、図示していないが、プライベート宣言された関数barから呼び出し元プログラムへリターンした時にも、DPの再設定等を行う必要はない。

【0199】

またこれまで説明した実施の形態では、例えば関数barの前に「private」と記述されるとプライベート関数であることが宣言され、DPの保存等の一連の処理が行われず、特に何も記述されなければプライベート関数でなく、DPの保存等の処理が行われ

る。

【0200】

しかし他の実施の形態では、例えば明示的にプライベート関数でないことを示すために「public」等の記述を行うことも可能である。

【0201】

このような記述は、「パブリック (public) 宣言」或いは「パブリック (public) 関数宣言」と呼ばれることもあり、この記述を行うことによって明示的にDPの保存等の一連の処理を行うことを示すことができる。

【0202】

また他の実施の形態では、例えばこの「プライベート宣言」が行われる関数は、同一プログラムモジュール内でのみ呼び出される、或いは呼び出すことが可能な関数であっても良く、更に他の実施の形態では、この「プライベート宣言」が行われる関数は、異なるプログラムモジュール間でのみ呼び出される、或いは呼び出すことが可能な関数であっても良い。

【0203】

尚、この例では新規キーワード「private」や「public」を追加し、C言語を拡張して宣言を行っているが、プラグマ等を用いることにより、言語使用を拡張することなく、宣言を行ってもよい。また、別ファイルとして記述したり、コンパイラの引数として指定してもよい。

【0204】

このように、アプリケーションプログラムモジュールに含まれるプログラムコード領域から呼び出されるライブラリプログラムモジュールのプログラムコード領域の記述に基づいて、処理を短縮するか否かが判別されるので、必ずしもアプリケーションプログラムモジュールとライブラリプログラムモジュールとが異なるデータ領域を使用することが必要でないケースでは、アドレスの保存や設定や再設定の処理を省略し、オブジェクトプログラムの実行を高速化することができ、不要な処理を出力しないことでオブジェクトプログラムのサイズを小さくすることができる。

【0205】

また、ライブラリプログラムモジュールのプログラムコード領域の記述が、当該プログラムコード領域毎に記述される宣言であるので、この宣言の有無により、ライブラリプログラムモジュールのプログラムコード領域の処理内容に応じてアドレスの保存や設定や再設定の処理を省略することができ、オブジェクトプログラムの実行を高速化することができる。

【0206】

(第3の実施の形態)

前記本発明の第1の実施の形態では、関数barが使用する全ての外部変数(例えばC)に対してアクセスするに際し、DPに保存されている値(例えば、0x800)に、DPからの相対アドレス(例えば、0x50)を加算したアドレス(例えば、0x850)でアクセスしている。

【0207】

その結果、関数barは、使用する全ての外部変数用領域として、アドレス例えば、0x800から始まるメモリ領域であるアプリケーションプログラムモジュール121用固有データ領域155を使用することができる。

【0208】

しかし次に説明する本発明の第3の実施の形態では、例えば関数barを呼び出す複数のアプリケーションプログラムモジュールが共通にアクセスするデータメモリ領域に関する記述を、例えば関数barの原始プログラムはその変数宣言部に含むことができる。

【0209】

図14は、本発明の第3の実施の形態におけるライブラリプログラムモジュールのソースプログラム及びオブジェクトプログラムの一例を示す図であり、図14(a)は、第3

の実施の形態におけるライブラリプログラムモジュールのソースプログラムの一例を示す図であり、図14(b)は、第3の実施の形態におけるライブラリプログラムモジュールのオブジェクトプログラムの一例を示す図である。

【0210】

図14では、C言語を例として宣言を記述している。この例ではC言語の言語仕様を拡張して新規のキーワード「common」を追加し、ソースコード上でこの「common」修飾詞をつけることにより宣言を記述している。

【0211】

例えば、図14(a)に示す関数barのプログラムでは、関数barが使用する外部変数としてDとCを使用する。その宣言が例えば図14(a)の「common int D;」と「int C;」の記述である。

【0212】

そして更に「common int D;」の記述は、この外部変数Dはこの関数barの複数の呼び出し元アプリケーションプログラムが共通にアクセスする外部変数であることを宣言している。

【0213】

尚、この例では新規のキーワード「common」を追加し、C言語を拡張して宣言を行っているが、プリAGMA等を用いて言語仕様を拡張することなく宣言を行ってもよい。

【0214】

また、別ファイル等に記述したり、コンパイラの引数として指定することによって宣言を行ってもよい。

【0215】

図14(a)に示すように、ライブラリモジュールプログラムのソースプログラムに含まれる外部変数Dには、この外部変数Dが、「common」である旨の記述が含まれている。この記述は、「外部変数Dがいわゆる共通外部変数である」旨を宣言する記述であり、例えば「コモン(common)宣言」或いは「コモン(common)外部変数宣言」と呼ばれることもある。

【0216】

図15は、本発明の第3の実施の形態におけるライブラリプログラムモジュールのコード領域及びデータ領域を示す図である。

【0217】

図15(a)は、プログラムコード領域553が、再入可能な関数barに相当し、このプログラムから固有データ領域であるアプリケーションプログラムモジュール121用データ領域555と、共有データ領域である共有データ領域559との両方がアクセスされ得ることを示している。尚ここでは共有データ領域としたが、必ずしも厳密に他の何らかの目的や用途と共有されていることは必要でなく、たまたま他の何らかの目的や用途と共有されていないものであってもかまわない。このことは他の説明についても同様である。

。

【0218】

図15(b)は、より具体的に、外部変数Cは固有データ領域であるアプリケーションプログラムモジュール121用データ領域555に格納され、外部変数Dは共有データ領域アクセスによってアクセスされる共有データ領域559に格納されることを示している。

。

【0219】

図14(a)の「int C;」の記述は、この外部変数Cは、この関数barの複数の呼び出し元アプリケーションプログラム毎に固有のメモリ領域を使用することを宣言している記述である。この記述によって、図15(b)に示すように、外部変数Cは固有データ領域アクセスによってアクセスされるアプリケーションプログラムモジュール(この例ではプログラムモジュール121)用の固有のデータ領域555に格納される。

【0220】

そして関数 `bar` では、呼び出し元アプリケーションプログラム（例えばプログラムモジュール 121）毎に固有のデータ領域 555 に格納されている外部変数 `C` に対してアクセスし、例えばそのメモリ領域に、値 10 を代入するに際し、`DP` に保存されている値（例えば、`0x800`）に、`DP` からの相対アドレス（例えば、`0x50`）を加算したアドレス（例えば、`0x850`）に、値 10 を代入する。この命令形式を図 14（b）と図 15（b）とに示す。この時の命令形式は前記第 1 の実施の形態と同一である。

【0221】

図 15（b）に示すように、関数 `bar` は、例えばアドレス `0x800` から始まるメモリ領域であるアプリケーションプログラムモジュール 121 用の固有データ領域 555 に対してアクセスすることができる。また、関数 `bar` に複数のアプリケーションプログラムモジュール（例えば、図 4（a）のアプリケーションプログラムモジュール 121 とアプリケーションプログラムモジュール 131）から同時に呼び出され、その結果、再入が起こっても外部変数へのアクセスが衝突することはない。

【0222】

図 14（a）の「`common int D;`」の記述は、この外部変数 `D` は、関数 `bar` の複数の呼び出し元アプリケーションプログラムが共有するメモリ領域を使用することを宣言している記述である。この記述によって、図 15（b）に示すように、外部変数 `D` は共有データ領域アクセスによってアクセスされる複数のアプリケーションプログラムモジュールが共通に使用する共有データ領域 559 に格納される。

【0223】

そして関数 `bar` では、呼び出し元アプリケーションプログラム（例えばプログラムモジュール 121）が共通に使用する共有データ領域 559 に格納されている外部変数 `D` に対してアクセスし、例えばそのメモリ領域に、値 20 を代入するに際し、特別なレジスタ `SDP` に保存されている値（例えば、`0xF00`）に、`SDP` からの相対アドレス（例えば、`0x60`）を加算したアドレス（例えば、`0xF60`）に、値 20 を代入する。この命令形式を図 14（b）と図 15（b）とに示す。

【0224】

この `SDP` は特別なレジスタであり、全アプリケーションプログラムに対して 1 つの共有データ領域の先頭アドレスを保存している。またこの `SDP` に保存されている値である全アプリケーションプログラムに対して 1 つの共有データ領域の先頭アドレス自体を `SDP` と呼ぶこともある。

【0225】

或いは他の実施の形態ではこの `SDP` として必ずしも特別なレジスタを使用せず、汎用的なレジスタを使用することもできる。このような実施の形態は汎用のレジスタを多く備えているような `MPU`（Micro Processing Unit）において特に有効であり、更にこの汎用レジスタがメモリアクセスのためのオフセットアドレスとして使用できる `MPU` では更に有効である。

【0226】

或いはまた他の実施の形態ではこの `SDP` として必ずしも特別なレジスタや汎用のレジスタを使用せず、通常のメモリやスタックを使用することもできる。

【0227】

しかしこのような実施の形態ではこのスタックや通常のメモリに格納された `SDP` をオフセットアドレスとして使用してメモリアクセスするために、幾らかの処理が必要になることもある。

【0228】

例えばその処理とは、スタックやメモリに格納されている `SDP` を読み出し、メモリアクセスのためのオフセットアドレスとして使用できるレジスタに格納したり、そのための準備として現在のレジスタの内容を退避する処理等である。

【0229】

これらのような方法で、この `SDP` レジスタが記憶している値をオフセットアドレスと

して使用してメモリ領域アクセスすることによって、図7 (b) に示すように、関数 `bar` は、例えばアドレス `0xF00` から始まるメモリ領域であるアプリケーションプログラムモジュール共有データ領域 `559` に対してアクセスすることができる。

【0230】

SDPの値は図示していないが、例えば図7 (a) に示すように、ライブラリプログラムモジュール `151` の自分固有データ領域アドレスと同様に固有データ領域 `125` に記憶しておき、アプリケーションプログラムモジュール `121` からライブラリプログラムモジュール `151` を呼び出す時、その呼び出す処理の中で保存と設定を行うこともできる。

【0231】

例えば、図3のステップ `S1` で現在のDP (`0x0600`) をスタックに保存する処理の前の処理として、現在のSDPをスタックに保存する処理を含めることも可能である。

【0232】

また図3のステップ `S2` で新しいDP (`0x0880`) を設定する処理の前の処理として、新しいSDP (`0x0F00`) を設定する処理を含めることも可能である。

【0233】

このように現在のSDPをスタックに保存する処理を、引数Aや引数Bをスタックに保存する処理よりも先に行うことによって、呼び出されるライブラリプログラムモジュールの側でこのスタックに保存されたSDPの有無に関与しなくてもよいことは、前記第1の実施の形態と同じであるから詳細な説明を省略する。

【0234】

このような実施の形態によって、関数 `bar` が、複数のアプリケーションプログラムモジュール (例えば、図4 (a) のアプリケーションプログラムモジュール `121` とアプリケーションプログラムモジュール `131`) から同時に呼び出され、その結果再入が起こった時、明示的に、その外部変数の幾つかを共通に使用し、データの共有、受け渡し、その他の処理を行うことができる。

【0235】

またこの時、前記したセマフォ等を使った調停が行われることがある、或いは行わなければならないことも発生し得るが、本発明の内容と直接関係しないので説明を省略する。

【0236】

このように、ライブラリプログラムモジュールが複数のアプリケーションプログラムモジュールから呼び出されたとしても、各アプリケーションプログラムモジュール毎に設けられた実行固有データメモリ領域にアクセスするため、例えば外部変数へのアクセスで衝突が発生することは起こらず、プログラムモジュールを再入可能に構成することができる。また、ライブラリプログラムモジュールが複数のアプリケーションプログラムモジュールから呼び出されたとしても、各アプリケーションプログラムモジュール毎で共有する実行共有データメモリ領域にアクセスするため、例えば外部変数の幾つかを共通に利用することができる。

【0237】

また、ライブラリプログラムモジュール用の実行共有データメモリ領域、及びライブラリプログラムモジュール用の実行固有データメモリ領域のうちのいずれを使用するのかを特定するための記述が、ライブラリプログラムモジュールに含まれる外部変数毎に記述されるコモン宣言であるので、このコモン宣言の有無を判断することによって容易に実行共有データメモリ領域と実行固有データメモリ領域とを使い分けることができる。

【0238】

(第4の実施の形態)

図16は、本発明の第4の実施の形態であるコンパイラが変換の対象とする呼び出し元プログラムであるアプリケーションプログラムモジュール `121` と、呼び出されるプログラムであるライブラリプログラムモジュール `151` と、その呼び出し処理時に行われる処理の一例を概念的に示す図である。図16 (a) は、各プログラムモジュールの固有のデータ領域へのアドレスを保存しているテーブルの一例を示す図であり、図16 (b) は、

第4の実施の形態におけるアプリケーションプログラムモジュールのコード領域及びデータ領域の一例を示す図であり、図16(c)は、第4の実施の形態におけるライブラリプログラムモジュールのコード領域及びデータ領域の一例を示す図である。

【0239】

図16(a)に示すように、第4の実施の形態では、呼び出し元アプリケーションプログラムモジュール単位、或いは呼び出し元アプリケーションプログラム単位、或いは呼び出し元アプリケーションプロセス単位、或いは呼び出し元アプリケーションタスク単位、或いは呼び出し元アプリケーション単位、或いは呼び出し元のプロセス単位に、或いはまた呼び出し元のタスク単位に、CCT (Context Control Table) と呼ばれる、各プログラムモジュールの固有のデータ領域へのアドレスを保存しているテーブルを備えている。

【0240】

呼び出し元のアプリケーションプログラムモジュール121では、アプリケーションプログラム121用CCTを参照してアプリケーションプログラムモジュール121自体のデータ領域アドレスである「0x600」を取り出し、この値をDPに設定し、この値をオフセットアドレスとして使用して外部変数アクセスすることによって、アプリケーションプログラムモジュール121用の固有データ領域125を使用する。

【0241】

このアプリケーションプログラムモジュール121からライブラリプログラムモジュール151に含まれる関数barを呼び出すには、まず関数barに引き渡す引数Aと引数Bをスタックに保存する。

【0242】

次に、アプリケーションプログラムモジュール121ではこの関数barから自分自身へ帰ってくるためのアドレス、即ち関数barへのジャンプ命令の次のアドレスをスタックに保存し、関数barに制御を移行する。

【0243】

制御の移行を受けた呼び出し先ライブラリプログラムモジュール151に含まれる関数barでは、現行のDPの内容を保存する。現行のDPの内容とは呼び出し元のアプリケーションプログラムモジュール121用の固有データ領域アドレスの値であり、関数barから呼び出し元のプログラムに戻った時、再度この値をDPに設定してアプリケーションプログラムモジュール121用の固有データ領域を使用するために保存しておくものである。

【0244】

次にこのDPの内容を参照してアプリケーションプログラムモジュール121用CCTへのポインタアドレスを取り出し、更にそのポインタアドレスを参照してアプリケーションプログラムモジュール121用CCTを取り出し、更にそのCCTの内容を参照して、関数barでアプリケーションプログラムモジュール121が固有に使用するアプリケーションプログラムモジュール121用データ領域155のアドレス（この例では、0x800）を取り出し、その内容をDPレジスタに設定する。

【0245】

アプリケーションプログラムモジュール121によって呼び出されたライブラリプログラムモジュール151に含まれる関数barでは、外部変数に対してアクセスする時、DPレジスタの値を相対アドレスのベースアドレスとして使用してアクセスするので、関数barが使用する全ての外部変数は0x800が相対アドレスのベースアドレスとして使用されてアクセスされ、アプリケーションプログラムモジュール121から呼び出される限りアプリケーションプログラムモジュール121用データ領域155が割り当てられる。

【0246】

これによって、複数のアプリケーションプログラムモジュールから関数barに対して再入が発生したとしても、外部変数で衝突が発生することは起こらない。

【0247】

関数 `bar` から呼び出し元のアプリケーションプログラムモジュール 121 に戻る時には、DP の内容を保存しておいた呼び出し元のアプリケーションプログラムモジュール 121 用の固有データ領域アドレスの値に戻してから、リターンする。

【0248】

これによって、呼び出し元のアプリケーションプログラムモジュール 121 では再度、呼び出し元のアプリケーションプログラムモジュール 121 自体のデータ領域を使用することができる。

【0249】

第 4 の実施の形態では、ライブラリプログラムモジュール 151 の呼び出し側で DP の設定を行わないため、呼び出し側のオブジェクトプログラムのサイズを小さくすることができ、呼び出し回数が多い場合等に特に有効である。

【0250】

このように、アプリケーションプログラムモジュールからライブラリプログラムモジュールへの再入が発生したとしても、各プログラムモジュール用データメモリ領域の先頭アドレスを記憶するテーブル (CCT) を介してアクセスするので、例えば外部変数へのアクセスで衝突が発生することは起こらず、プログラムモジュールを再入可能に構成することができる。

【0251】

また図示していないが、図 16 (c) で関数 `bar` が「`private`」宣言が行われていれば、上記のような DP の保存やアプリケーションプログラムモジュール 121 からの呼び出しで使用する関数 `bar` 用の DP を、CCT から取り出して設定等の処理を行わないことは上記第 2 の実施の形態と同じであり、「`public`」宣言が行われているならば明示的に、DP の保存やアプリケーションプログラムモジュール 121 からの呼び出しで使用する関数 `bar` 用の DP を CCT から取り出して設定等の処理等が行われることを示す。これらは前記第 2 の実施の形態と同じであるから説明は省略する。

【0252】

この場合、アプリケーションプログラムモジュールに含まれるプログラムコード領域から呼び出されるライブラリプログラムモジュールに含まれるプログラムコード領域の記述に基づいて、処理を短縮するか否かが判別される。したがって、必ずしもアプリケーションプログラムモジュールと呼び出されるライブラリプログラムモジュールとが異なるデータ領域を使用することが必要でないケースでは、各プログラムモジュール用データ領域の先頭アドレスを記憶したテーブル (CCT) を介した先頭アドレスの設定の処理を省略し、オブジェクトプログラムの実行を高速化することができ、不要な処理を出力しないことでオブジェクトプログラムのサイズを小さくすることができる。

【0253】

更にまた図示していないが、この CCT には前記 DP だけでなく、前記第 3 の実施の形態で説明した新しい SDP (0x0F00) を、図 7 (a) の固有データ領域 125 に格納して呼び出しプログラムモジュール 121 のプログラムコード領域 123 でその保存や設定を行うのではなく、例えば、図 16 (a) の CCT と同様のテーブルを用意して、そのテーブルに各プログラムモジュールの共有データ領域の先頭アドレスを格納しておき、呼び出されるライブラリプログラムモジュールである、例えば図 16 (c) のライブラリプログラムモジュール 151 のプログラムコード領域 153 における処理の中で、その CCT と同様のテーブルと、CCT と同様のテーブルに格納された SDP とを読み出して設定することもできる。これについても前記 DP の保存と設定と同じであるから詳細な説明は省略する。また、CCT と同様のテーブルは、CCT と異なり、実行環境で 1 つだけ保持するとしてもよい。

【0254】

(第 5 の実施の形態)

次に、本発明の第 5 の実施の形態について説明する。本発明の第 5 の実施の形態では、プログラムモジュールに識別番号を付与し、当該識別番号と各プログラムモジュールのデ

ータ領域のDPとを対応付けたテーブルを作成し、アプリケーションプログラムモジュールからライブラリプログラムモジュールが呼び出されると、各プログラムモジュールの識別番号をインデックスとしてテーブルを参照して、ライブラリプログラムモジュールのデータ領域のDPと、アプリケーションプログラムモジュールのデータ領域のDPとを切り替える。

【0255】

図17は、本発明の第5の実施の形態におけるコンパイル装置の構成を示すブロック図である。図17に示すコンパイル装置14は、プログラム実行部16及びプログラム記憶部17を備えて構成される。

【0256】

プログラム実行部16は、例えばCPU（中央演算処理装置）等で構成され、アドレス保存プログラム生成部16a、アドレス設定プログラム生成部16b、移行プログラム生成部16c、アドレス再設定プログラム生成部16d、アクセスプログラム生成部16e、識別番号付与プログラム生成部16g、テーブル作成プログラム生成部16h、識別番号取得プログラム生成部16i及び先頭アドレス取得プログラム生成部16jを備えて構成される。プログラム記憶部17は、コンパイラプログラム記憶部17a、ソースプログラム記憶部17b及びオブジェクトプログラム記憶部17cを備えて構成される。

【0257】

なお、図17に示す第5の実施の形態におけるコンパイル装置と、図2に示す第1の実施の形態におけるコンパイル装置とで同じ構成物については、同じ符号を付し、以下の説明では第1の実施の形態とは異なる構成についてのみ説明する。

【0258】

識別番号付与プログラム生成部16gは、呼び出し元プログラムモジュールを識別する識別番号と、他プログラムモジュールを識別する識別番号とをそれぞれに付与するための識別番号付与プログラムを生成する。

【0259】

テーブル作成プログラム生成部16hは、付与された呼び出し元プログラムモジュールの識別番号と、呼び出し元プログラムモジュールのデータ領域の先頭アドレスとを対応付けるとともに、付与された他プログラムモジュールの識別番号と、他プログラムモジュールのデータ領域の先頭アドレスとを対応付けるテーブルを作成するためのテーブル作成プログラムを生成する。

【0260】

識別番号取得プログラム生成部16iは、呼び出し元プログラムモジュールから他プログラムモジュールへの呼び出し命令を受けて、他プログラムモジュールの識別番号を取得するための識別番号取得プログラムを生成する。

【0261】

先頭アドレス取得プログラム生成部16jは、取得された識別番号をインデックスとして、作成されたテーブルから他プログラムモジュールのデータ領域の先頭アドレスを取得するための先頭アドレス取得プログラムを生成する。

【0262】

図18は、本発明の第5の実施の形態における端末装置の構成を示すブロック図である。図18に示す端末装置12は、プログラム実行部18、メモリ19、データ読込部20及び通信部21を備えて構成される。通信部21は、サーバ11から送信されるオブジェクトプログラムをダウンロードするものであり、識別番号付与部21aを備えて構成される。データ読込部20は、通信部21によってダウンロードされたオブジェクトプログラムをメモリ19に読み込むものであり、DPテーブル作成部20aを備えて構成される。プログラム実行部18は、例えばDSP等で構成され、アドレス保存部18a、アドレス設定部18b、移行部18c、アドレス再設定部18d、アクセス部18e、識別番号取得部18g及び先頭アドレス取得部18hを備えて構成される。メモリ19は、アプリケーションプログラムモジュール記憶部19a、ライブラリプログラムモジュール記憶部1

9b、スタック19c及びDPテーブル記憶部19dを備えて構成される。

【0263】

なお、図18に示す第5の実施の形態における端末装置と、図5に示す第1の実施の形態における端末装置とで同じ構成物については、同じ符号を付し、以下の説明では第1の実施の形態とは異なる構成についてのみ説明する。

【0264】

識別番号付与部21aは、呼び出し元プログラムモジュールであるアプリケーションプログラムモジュールを識別する識別番号を当該アプリケーションプログラムモジュールに付与するとともに、他プログラムモジュールであるライブラリプログラムモジュールを識別する識別番号を当該ライブラリプログラムモジュールに付与する。

【0265】

DPテーブル作成部20aは、識別番号付与部21aによって付与されたアプリケーションプログラムモジュールの識別番号と、アプリケーションプログラムモジュールのデータ領域の先頭アドレス(DP)とを対応付けるとともに、識別番号付与部21aによって付与されたライブラリプログラムモジュールの識別番号と、ライブラリプログラムモジュールのデータ領域の先頭アドレスとを対応付けるDPテーブルを作成し、DPテーブル記憶部19dに記憶する。

【0266】

識別番号取得部18gは、アプリケーションプログラムモジュールからライブラリプログラムモジュールへの呼び出し命令を受けて、ライブラリプログラムモジュールの識別番号を取得する。

【0267】

先頭アドレス取得部18hは、識別番号取得部18gによって取得された識別番号をインデクスとして、DPテーブル作成部20aによって作成されたDPテーブルからライブラリプログラムモジュールのデータ領域の先頭アドレスを取得する。

【0268】

アドレス設定部18bは、アプリケーションプログラムモジュールによって呼び出されるライブラリプログラムモジュールが使用するライブラリプログラムモジュール用データ領域の先頭アドレスを、先頭アドレス取得部18hによって取得された先頭アドレスに設定する。

【0269】

DPテーブル記憶部19dは、DPテーブル作成部20aによって作成されたDPテーブルを記憶する。

【0270】

図19は、図18に示す端末装置による動的モジュールリンク処理の一例を示すフローチャートである。

【0271】

ステップS31において、通信部21は、サーバ11のコンパイル装置14によってソースプログラムからオブジェクトプログラムに変換されたアプリケーションプログラムモジュール及びライブラリプログラムモジュールをダウンロードする。

【0272】

ステップS32において、識別番号付与部21aは、ダウンロードされたアプリケーションプログラムモジュール及びライブラリプログラムモジュールのそれぞれに対して固有の識別番号を付与する。識別番号付与部21aによって付与された識別番号は、アプリケーションプログラムモジュール記憶部19a及びライブラリプログラムモジュール記憶部19bにそれぞれ記憶される。

【0273】

ステップS33において、データ読込部20は、アプリケーションプログラムモジュール及びライブラリプログラムモジュールのコード領域を読み込む。

【0274】

ステップS34において、DPテーブル作成部20aは、メモリ19のDPテーブル記憶部19dに、各プログラムモジュールのデータ領域の先頭アドレスと、各プログラムモジュールの識別番号とが対応付けられたDPテーブルを作成する。なお、この時点ではDPテーブルにはデータがまだ入力されておらず、メモリ内の領域のみが確保される。

【0275】

ステップS35において、データ読込部20は、アプリケーションプログラムモジュール及びライブラリプログラムモジュールにデータ領域を割り当てる。すなわち、アプリケーションプログラムモジュール及びライブラリプログラムモジュールのデータ領域を、メモリ19のアプリケーションプログラムモジュール記憶部19a及びライブラリプログラムモジュール記憶部19bにそれぞれ記憶する。

【0276】

ステップS36において、DPテーブル作成部20aは、アプリケーションプログラムモジュールのデータ領域の先頭アドレス(DP)をDPテーブル記憶部19dに記憶されているアプリケーションプログラムモジュールの識別番号に対応付けて設定し、ライブラリプログラムモジュールのデータ領域の先頭アドレスをDPテーブル記憶部19dに記憶されているライブラリプログラムモジュールの識別番号に対応付けて設定する。

【0277】

ステップS37において、識別番号取得部18gは、アプリケーションプログラムモジュールからライブラリプログラムモジュールが呼び出されたか否かを判断する。ここで、アプリケーションプログラムモジュールからライブラリプログラムモジュールが呼び出されたと判断されると、ステップS38に移行し、アプリケーションプログラムモジュールからライブラリプログラムモジュールが呼び出されていないと判断されると、アプリケーションプログラムモジュールからライブラリプログラムモジュールが呼び出されるまで待機状態となる。

【0278】

ステップS38において、アドレス保存部18aは、アプリケーションプログラムモジュールのコード領域の先頭アドレス及びデータ領域の先頭アドレスをメモリのスタック19cに保存することによって退避させる。

【0279】

ステップS39において、識別番号取得部18gは、ライブラリプログラムモジュール記憶部19bに記憶されているライブラリプログラムモジュールの識別番号を取得する。

【0280】

ステップS40において、先頭アドレス取得部18hは、識別番号取得部18gによって取得されたライブラリプログラムモジュールの識別番号をインデックスとしてDPテーブル記憶部19dに記憶されているDPテーブルを参照し、当該ライブラリプログラムモジュールの識別番号に対応する先頭アドレスを取得する。

【0281】

ステップS41において、アドレス設定部は、アプリケーションプログラムモジュールのデータ領域の先頭アドレスを、先頭アドレス取得部18hによって取得されたライブラリプログラムモジュールのデータ領域の先頭アドレスに設定することによって切り替える。

【0282】

ステップS42において、アドレス再設定部18dは、アドレス保存部によってメモリ19のスタック19cに退避させたアプリケーションプログラムモジュールのコード領域の先頭アドレス及びデータ領域の先頭アドレスを再設定することによって復元する。

【0283】

このように、まず、識別番号付与プログラム生成部16gによって、アプリケーションプログラムモジュールを識別する識別番号と、ライブラリプログラムモジュールを識別する識別番号とをそれぞれに付与するための識別番号付与プログラムが生成される。そして、テーブル作成プログラム生成部16hによって、付与されたアプリケーションプログラ

ムモジュールの識別番号と、アプリケーションプログラムモジュールのデータ領域の先頭アドレスとを対応付けるとともに、付与されたライブラリプログラムモジュールの識別番号と、ライブラリプログラムモジュールのデータ領域の先頭アドレスとを対応付けるテーブルを作成するためのテーブル作成プログラムが生成される。アプリケーションプログラムモジュールからライブラリプログラムモジュールへの呼び出し命令を受けて、識別番号取得プログラム生成部16iによって、ライブラリプログラムモジュールの識別番号を取得するための識別番号取得プログラムが生成される。先頭アドレス取得プログラム生成部16jによって、取得された識別番号をインデックスとして、作成されたテーブルからライブラリプログラムモジュールのデータ領域の先頭アドレスを取得するための先頭アドレス取得プログラムが生成される。さらに、アドレス設定プログラム生成部16bによって、取得されたライブラリプログラムモジュールのデータ領域の先頭アドレスと、アプリケーションプログラムモジュールのデータ領域の先頭アドレスとを切り替えるための先頭アドレス切替プログラムが生成される。

【0284】

したがって、従来のMMU (Memory Management Unit) のようにアドレス変換するためのテーブルをページ単位で作成するのではなく、モジュール単位で作成することによって、テーブルのデータ量を削減することができ、資源制約の厳しい小型の機器に適合させることができる。また、テーブルのデータ量を削減することができるので、従来のMMUによるアドレス変換に比して実行速度を向上させることができる。

【0285】

なお、コンパイル装置14のプログラム実行部16は、ライブラリプログラムモジュール外で定義された変数又は関数へのアドレスを保持するためのアドレス保持プログラムを生成するアドレス保持プログラム生成部を備え、端末装置12のプログラム実行部18は、ライブラリプログラムモジュール外で定義された変数又は関数へのアドレスを保持するアドレス保持部を備えてもよい。この場合、アクセス部18eは、ライブラリプログラムモジュール内で定義された変数へアクセスする場合、先頭アドレス取得部18hによって取得された先頭アドレスからの相対アドレスでアクセスし、ライブラリプログラムモジュール内で定義された関数へアクセスする場合、PC (プログラムカウンタ) からの相対アドレスでアクセスする。また、アクセス部18eは、ライブラリプログラムモジュール外で定義された変数又は関数へアクセスする場合、アドレス保持部によって保持されたアドレスに、先頭アドレス取得部18hによって取得された先頭アドレスからの相対アドレスでアクセスした後、アドレス保持部によって保持されたアドレス経由で間接アクセスする。

【0286】

この構成によれば、アドレス保持プログラム生成部によって、ライブラリプログラムモジュール外で定義された変数又は関数へのアドレスを保持するためのアドレス保持プログラムが生成される。そして、アクセスプログラム生成部16eによって、ライブラリプログラムモジュール内で定義された変数又は関数へアクセスする場合、取得された先頭アドレスからの相対アドレスでアクセスされ、ライブラリプログラムモジュール内で定義された関数へアクセスする場合、プログラムカウンタからの相対アドレスでアクセスされ、ライブラリプログラムモジュール外で定義された変数又は関数へアクセスする場合、保持されたアドレスに取得された先頭アドレスからの相対アドレスでアクセスした後、保持されたアドレス経由で間接アクセスするためのアクセスプログラムが生成される。したがって、ライブラリプログラムモジュール外で定義された変数又は関数へアクセスする場合に、保持されたアドレス経由で間接的にアクセスすることができる。

【0287】

なお、本実施の形態では、コンパイル装置14が備えるテーブル作成プログラム生成部16hによって生成されたテーブル作成プログラムを端末装置12がダウンロードすることによって、データ読込部20がDPテーブル作成部20aとして機能し、識別番号付与プログラム生成部16gによって生成された識別番号付与プログラムを端末装置12がダ

ウンロードすることによって、通信部 21 が識別番号付与部 21 a として機能している。しかしながら、あらかじめ、データ読込部 20 が DP テーブル作成部 20 a としての機能を備えていてもよく、通信部 21 が識別番号付与部 21 a としての機能を備えていてもよい。この場合、図 17 に示すコンパイル装置 14 のプログラム実行部 16 は、アドレス保存プログラム生成部 16 a、アドレス設定プログラム生成部 16 b、移行プログラム生成部 16 c、アドレス再設定プログラム生成部 16 d、アクセスプログラム生成部 16 e、識別番号取得プログラム生成部 16 i 及び先頭アドレス取得プログラム生成部 16 j のみを備えて構成される。

【0288】

このように、識別番号取得プログラム生成部 16 i によって、アプリケーションプログラムモジュールからライブラリプログラムモジュールへの呼び出し命令を受けて、ライブラリプログラムモジュールを識別する識別番号を取得するための識別番号取得プログラムが生成される。先頭アドレス取得プログラム生成部 16 j によって、取得された識別番号をインデックスとして、アプリケーションプログラムモジュールの識別番号と、アプリケーションプログラムモジュールのデータ領域の先頭アドレスとが対応付けられるとともに、ライブラリプログラムモジュールの識別番号と、ライブラリプログラムモジュールのデータ領域の先頭アドレスとが対応付けられたテーブルから、ライブラリプログラムモジュールのデータ領域の先頭アドレスを取得するための先頭アドレス取得プログラムが生成される。そして、アドレス再設定プログラム生成部 16 d によって、取得されたライブラリプログラムモジュールのデータ領域の先頭アドレスと、アプリケーションプログラムモジュールのデータ領域の先頭アドレスとを切り替えるための先頭アドレス切替プログラムが生成される。

【0289】

したがって、アドレス変換するためのテーブルを、ページ単位ではなくモジュール単位で行うことによって、テーブルのデータ量を削減することができ、資源制約の厳しい小型の機器に適合させることができる。また、識別番号を付与するプログラムやテーブルを作成するプログラムをコンパイルする必要がないため、コンパイル処理に要する時間を短縮することができ、さらに、端末装置 12 に送信するプログラムモジュールのデータ量を削減することができる。

【0290】

次に、メモリのフラグメンテーション（細分化）の解消方法について説明する。メモリに対して書き込みや削除を繰り返す行くと、メモリ内の空き領域が細分化されてしまい、1つのデータを連続してメモリに記憶させることが困難となる。この場合、データを複数の領域に分割して記憶させなければならず、メモリへのアクセス速度の低下に繋がる。そのため、細分化されたメモリ領域をひとまとめにし、連続したメモリ領域にする必要がある。

【0291】

図 20 は、フラグメンテーションを解消するためのコンパイル装置及び端末装置の構成を示すブロック図である。なお、他の実施の形態と同じ構成物については、同じ符号を付し、説明を省略する。

【0292】

図 20 に示すように、サーバ 11 は、通信部 22 及びコンパイル装置 14 を備えて構成される。通信部 22 は、コンパイル装置 14 によって変換されたオブジェクトプログラムを端末装置 12 に送信する。コンパイル装置 14 は、プログラム実行部 16 及びプログラム記憶部 17 を備えて構成される。プログラム実行部 16 は、アプリケーション終了プログラム生成部 23 a、コンパクションプログラム生成部 23 b 及びアプリケーション再開プログラム生成部 23 c を備えて構成される。

【0293】

アプリケーション終了プログラム生成部 23 a は、少なくとも 1つのプログラムモジュールで構成されるとともに、現在実行中であるアプリケーションを終了するための終了プ

ログラムを生成する。

【0294】

コンパクションプログラム生成部23bは、アプリケーションが終了された後、メモリ19をコンパクションするためのコンパクションプログラムを生成する。なお、コンパクションとは、細かく分断されたメモリ19内の領域をひとまとめにし、それらを連続した領域にすることである。

【0295】

アプリケーション再開プログラム生成部23cは、メモリ19がコンパクションされた後、終了されたアプリケーションを最初から再開するための再開プログラムを生成する。

【0296】

端末装置12は、プログラム実行部18、メモリ19及び通信部21を備えて構成される。通信部21は、サーバ11より送信されたオブジェクトプログラムを受信し、メモリ19に記憶する。プログラム実行部18は、アプリケーション終了部24a、コンパクション部24b及びアプリケーション再開部24cを備えて構成される。

【0297】

アプリケーション終了部24aは、少なくとも1つのプログラムモジュールで構成されるとともに、現在実行中であるアプリケーションを終了する。コンパクション部24bは、アプリケーション終了部24aによってアプリケーションが終了された後、メモリ19をコンパクションする。アプリケーション再開部24cは、メモリ19がコンパクションされた後、アプリケーション終了部24aによって終了されたアプリケーションを最初から再開する。

【0298】

このように、アプリケーション終了プログラム生成部23aによって、少なくとも1つのモジュールで構成され、かつ現在実行中であるアプリケーションを終了するための終了プログラムが生成される。そして、アプリケーションが終了された後、コンパクションプログラム生成部23bによって、メモリ19をコンパクションするためのコンパクションプログラムが生成され、メモリ19がコンパクションされた後、アプリケーション再開プログラム生成部23cによって、アプリケーションを最初から再開するための再開プログラムが生成される。したがって、メモリのフラグメンテーションを解消することができ、メモリを有効的に利用することができる。

【0299】

次に、メモリのフラグメンテーションを解消する第1の変形例について説明する。第1の変形例では、アドレス値に依存しないような情報で現在のアプリケーションの状態を保存し、コンパクション後、その状態まで実行させるものである。

【0300】

図21は、第1の変形例におけるフラグメンテーションを解消するためのコンパイル装置及び端末装置の構成を示すブロック図である。なお、他の実施の形態と同じ構成物については、同じ符号を付し、説明を省略する。

【0301】

図21に示すように、サーバ11は、通信部22及びコンパイル装置14を備えて構成される。コンパイル装置14は、プログラム実行部16及びプログラム記憶部17を備えて構成される。プログラム実行部16は、アプリケーション終了プログラム生成部23a、アプリケーション情報保存プログラム生成部23d、コンパクションプログラム生成部23b及びアプリケーション実行プログラム生成部23eを備えて構成される。

【0302】

アプリケーション情報保存プログラム生成部23dは、終了されるまでのアプリケーションのアドレス値に依存しない情報を保存するための保存プログラムを生成する。

【0303】

アプリケーション実行プログラム生成部23eは、メモリ19がコンパクションされた後、保存された情報を読み出し、読み出された情報に基づいてアプリケーションが終了し

た時点における状態まで当該アプリケーションを実行するための実行プログラムを生成する。

【0304】

端末装置12は、プログラム実行部18、メモリ19及び通信部21を備えて構成される。プログラム実行部18は、アプリケーション終了部24a、アプリケーション情報保存部24d、コンパクション部24b及びアプリケーション実行部24eを備えて構成される。

【0305】

アプリケーション情報保存部24dは、アプリケーション終了部24aによって終了されるまでのアプリケーションのアドレス値に依存しない情報を保存する。アプリケーション実行部24eは、コンパクション部24bによってメモリ19がコンパクションされた後、アプリケーション情報保存部24dによって保存された情報を読み出し、読み出された情報に基づいてアプリケーションが終了した時点における状態まで当該アプリケーションを実行する。

【0306】

このように、アプリケーション終了プログラム生成部23aによって、少なくとも1つのモジュールで構成され、かつ現在実行中であるアプリケーションを終了するためのアプリケーション終了プログラムが生成される。そして、アプリケーション情報保存プログラム生成部23dによって、終了されるまでのアプリケーションのアドレス値に依存しない情報を保存するためのアプリケーション情報保存プログラムが生成される。コンパクションプログラム生成部23bによって、メモリ19をコンパクションするためのコンパクションプログラムが生成される。メモリ19がコンパクションされた後、アプリケーション実行プログラム生成部23eによって、保存された情報を読み出し、読み出された情報に基づいてアプリケーションが終了した時点における状態まで当該アプリケーションを実行するためのアプリケーション実行プログラムが生成される。したがって、メモリのフラグメンテーションを解消することができ、メモリを有効的に利用することができる。

【0307】

次に、メモリのフラグメンテーションを解消する第2の変形例について説明する。第2の変形例では、全メモリ空間に対してリファレンスフラグ（識別子）の領域を設け、メモリアクセスの際に、アドレス値を設定する場合と、数値等のアドレス値以外を設定する場合とをリファレンスフラグによって識別し、コンパクション後、リファレンスフラグからアドレス値が設定されているエントリについてリロケーション（再配置）を行うものである。

【0308】

図22は、第2の変形例におけるフラグメンテーションを解消するためのコンパイル装置及び端末装置の構成を示すブロック図である。なお、他の実施の形態と同じ構成物については、同じ符号を付し、説明を省略する。

【0309】

図22に示すように、サーバ11は、通信部22及びコンパイル装置14を備えて構成される。コンパイル装置14は、プログラム実行部16及びプログラム記憶部17を備えて構成される。プログラム実行部16は、メモリ状態記憶プログラム生成部23f、コンパクションプログラム生成部23b及び再配置プログラム生成部23gを備えて構成される。

【0310】

メモリ状態記憶プログラム生成部23fは、コンパクション前のメモリ19の状態（メモリマップ）を記憶するためのメモリ状態記憶プログラムを生成する。

【0311】

再配置プログラム生成部23gは、メモリ19がコンパクションされた後、アドレス値を設定する場合と、数値等のアドレス値以外を設定する場合とを識別するリファレンスフラグに基づいてアドレス値が設定されているエントリに対して、記憶されているコンパク

ション前のメモリ19の状態を参照して、アドレス値の再配置を行うための再配置プログラムを生成する。

【0312】

端末装置12は、プログラム実行部18、メモリ19及び通信部21を備えて構成される。プログラム実行部18は、メモリ状態記憶部24f、コンパクション部24b及び再配置部24gを備えて構成される。

【0313】

メモリ状態記憶部24fは、コンパクション前のメモリ19の状態（メモリマップ）を記憶する。再配置部24gは、メモリ19がコンパクションされた後、アドレス値を設定する場合と、数値等のアドレス値以外を設定する場合とを識別するリファレンスフラグに基づいてアドレス値が設定されているエントリに対して、記憶されているコンパクション前のメモリ19の状態を参照して、アドレス値の再配置を行う。

【0314】

図23は、第2の変形例におけるフラグメンテーションの解消方法について説明するための図であり、図23(a)は、領域が細分化されたコンパクション前のメモリの状態を示す図であり、図23(b)は、コンパクション後のメモリの状態を示す図であり、図23(c)は、リロケーション後のメモリの状態を示す図である。

【0315】

図23(a)に示すメモリ19は、アドレスが0x300から0x400までの領域が未使用となっており、0x550にアドレス値である0x558が格納されており、0x554に3という数値が格納されており、0x558に100という数値が格納されており、0x700から未使用の領域となっており、その他の領域には他のデータが格納されている。本変形例では、メモリ19の全メモリ空間に対してリファレンスフラグの領域RFが設けられている。この領域RFには、アドレス値が設定される場合は「1」が設定され、数値等のアドレス値以外が設定される場合は「0」が設定される。図23(a)では、アドレス0x550には、ポインタ変数としてアドレス値0x558が設定されているので、リファレンスフラグが1に設定されており、アドレス0x554及び0x558には、数値3及び100が設定されているので、リファレンスフラグが0に設定されている。

【0316】

図23(a)に示す状態のメモリ19をコンパクションすると、図23(b)に示す状態となる。すなわち、図23(b)に示すように、メモリ19は、未使用の領域であった0x300から0x400までの領域がなくなり、0x100から0x600までの領域がひとまとまりとなり、0x600から未使用の領域となっている。この場合、メモリ19内のアドレスがずれるため、コンパクション前にアドレス値0x558を格納していたアドレス0x550は0x450に移動し、アドレス0x550がポイントしていたアドレス0x558も0x458に移動している。この結果、図23(b)に示すように、アドレス0x450に格納されているポインタ変数がポイントするアドレスがなくなってしまう。

【0317】

そこで、メモリ状態記憶部24fは、コンパクション前のメモリ19の状態を記憶しておき、再配置部24gは、メモリ19がコンパクションされた後、リファレンスフラグの値を識別し、当該リファレンスフラグが1に設定されているエントリに対して、記憶されているコンパクション前のメモリの状態とコンパクション後のメモリ19の状態とを参照して、アドレス値の再配置を行う。例えば、図23(c)に示すように、アドレス0x450に格納されているアドレス値は、0x558から正しいアドレス値である0x458に再配置される。

【0318】

このように、メモリ19内において、アドレス値を設定するか否かを示す識別子を設け、メモリ状態記憶プログラム生成部23fによって、コンパクション前のメモリ19の状

態を記憶するためのメモリ状態記憶プログラムが生成される。そして、コンパクションプログラム生成部23bによって、メモリ19をコンパクションするためのコンパクションプログラムが生成される。さらに、メモリ19がコンパクションされた後、再配置プログラム生成部23gによって、識別子に基づいてアドレス値が設定されているエントリに対して、記憶されているコンパクション前のメモリ19の状態を参照して、アドレス値の再配置を行うための再配置プログラムが生成される。したがって、メモリのフラグメンテーションを解消することができ、メモリを有効的に利用することができる。

【0319】

次に、メモリのフラグメンテーションを解消する第3の変形例について説明する。第3の変形例では、メモリ内の領域を、アドレス値が格納されるアドレス値用メモリ領域と、数値等のアドレス値以外が格納される非アドレス値用メモリ領域とに分割し、コンパクション後、アドレス値用メモリ領域のみについてアドレス値のリロケーション（再配置）を行うものである。

【0320】

図24は、第3の変形例におけるフラグメンテーションを解消するためのコンパイル装置及び端末装置の構成を示すブロック図である。なお、他の実施の形態と同じ構成物については、同じ符号を付し、説明を省略する。

【0321】

図24に示すように、サーバ11は、通信部22及びコンパイル装置14を備えて構成される。コンパイル装置14は、プログラム実行部16及びプログラム記憶部17を備えて構成される。プログラム実行部16は、メモリ状態記憶プログラム生成部23h、コンパクションプログラム生成部23b及びアドレス値用メモリ再配置プログラム生成部23iを備えて構成される。

【0322】

メモリ状態記憶プログラム生成部23hは、アドレス値を格納するアドレス値用メモリ領域及びアドレス値以外を格納する非アドレス値用メモリ領域のコンパクション前の状態を記憶するためのメモリ状態記憶プログラムを生成する。

【0323】

アドレス値用メモリ再配置プログラム生成部23iは、アドレス値用メモリ領域及び非アドレス値用メモリ領域がコンパクションされた後、記憶されているコンパクション前のアドレス値用メモリ領域の状態を参照して、アドレス値用メモリ領域のみのアドレス値の再配置を行うための再配置プログラムを生成する。

【0324】

端末装置12は、プログラム実行部18、メモリ19及び通信部21を備えて構成される。プログラム実行部18は、メモリ状態記憶部24h、コンパクション部24b及びアドレス値用メモリ再配置部24iを備えて構成される。

【0325】

メモリ19は、アドレス値を格納するアドレス値用メモリ領域と、数値等のアドレス値以外を格納する非アドレス値用メモリ領域に分割されている。

【0326】

メモリ状態記憶部24hは、アドレス値を格納するアドレス値用メモリ領域及びアドレス値以外を格納する非アドレス値用メモリ領域のコンパクション前の状態を記憶する。

【0327】

アドレス値用メモリ再配置部24iは、アドレス値用メモリ領域及びアドレス値以外を格納する非アドレス値用メモリ領域がコンパクションされた後、メモリ状態記憶部24hによって記憶されているコンパクション前のアドレス値用メモリ領域の状態を参照して、アドレス値用メモリ領域のみのアドレス値の再配置を行う。

【0328】

図25は、第3の変形例におけるフラグメンテーションの解消方法について説明するための図であり、図25(a)は、領域が細分化されたコンパクション前のメモリの状態を

示す図であり、図 25 (b) は、コンパクション後のメモリの状態を示す図であり、図 25 (c) は、リロケーション後のメモリの状態を示す図である。

【0329】

図 25 (a) に示すように、メモリ 19 は、アドレス値を格納しているアドレス値用メモリ領域と、数値等のアドレス値以外を格納している非アドレス値用メモリ領域とに分割されている。

【0330】

図 25 (a) に示すメモリ 19 のアドレス値用メモリ領域は、アドレスが 0×100 から 0×200 までの領域が未使用となっており、 0×300 にアドレス値である 0×304 が格納されており、 0×304 に 0×200 が格納されており、 0×500 から未使用の領域となっており、その他の領域には他のデータが格納されている。また、メモリ 19 の非アドレス値用メモリ領域は、アドレスが 0×5000 から 0×5100 までの領域が未使用となっており、 0×5200 に 1000 という数値が格納されており、 0×5204 に 2000 という数値が格納されており、 0×5400 から未使用の領域となっており、その他の領域には他のデータが格納されている。

【0331】

図 25 (a) に示す状態のメモリ 19 をコンパクションすると、図 25 (b) に示す状態となる。すなわち、図 25 (b) に示すように、メモリ 19 のアドレス値用メモリ領域は、未使用の領域であった 0×100 から 0×200 までの領域がなくなり、 0×100 から 0×400 までの領域がひとまとまりとなり、 0×400 から未使用の領域となっている。また、メモリ 19 の非アドレス値用メモリ領域は、未使用の領域であった 0×5000 から 0×5100 までの領域がなくなり、 0×5000 から 0×5300 までの領域がひとまとまりとなり、 0×5300 から未使用の領域となっている。

【0332】

この場合、メモリ 19 内のアドレスがずれるため、コンパクション前にアドレス値 0×304 を格納していたアドレス 0×300 は 0×200 に移動し、コンパクション前にアドレス値 0×200 を格納していたアドレス 0×304 は 0×204 に移動し、アドレス 0×304 がポイントしていたアドレス 0×200 も 0×100 に移動している。この結果、図 25 (b) に示すように、アドレス 0×200 及び 0×204 に格納されているポインタ変数がポイントするアドレスがなくなってしまう。

【0333】

そこで、メモリ状態記憶部 24 h は、メモリ 19 のアドレス値用メモリ領域及び非アドレス値用メモリ領域のコンパクション前の状態を記憶しておき、アドレス値用メモリ再配置部 24 i は、メモリ 19 がコンパクションされた後、メモリ状態記憶部 24 h によって記憶されているコンパクション前のメモリ 19 のアドレス値用メモリ領域の状態とコンパクション後のメモリ 19 のアドレス値用メモリ領域の状態とを参照して、アドレス値用メモリ領域のみのアドレス値の再配置を行う。例えば、図 25 (c) に示すように、アドレス値用メモリ再配置部 24 i によって、アドレス 0×200 に格納されているアドレス値は、 0×304 から正しいアドレス値である 0×204 に再配置され、アドレス 0×204 に格納されているアドレス値は、 0×200 から正しいアドレス値である 0×100 に再配置される。

【0334】

このように、アドレス値を格納するアドレス値用メモリ領域と、アドレス値以外を格納する非アドレス値用メモリ領域とを設け、メモリ状態記憶プログラム生成部 23 h によって、コンパクション前のアドレス値用メモリ領域及び非アドレス値用メモリ領域の状態を記憶するためのメモリ状態記憶プログラムが生成される。コンパクションプログラム生成部 23 b によって、アドレス値用メモリ領域及び非アドレス値用メモリ領域をコンパクションするためのコンパクションプログラムが生成される。アドレス値用メモリ領域及び非アドレス値用メモリ領域がコンパクションされた後、アドレス値用メモリ再配置プログラム生成部 23 i によって、記憶されているコンパクション前のアドレス値用メモリ領域の

状態を参照して、アドレス値用メモリ領域のみのアドレス値の再配置を行うためのアドレス値用メモリ再配置プログラムが生成される。したがって、メモリのフラグメンテーションを解消することができ、メモリを有効的に利用することができる。

【0335】

次に、メモリのフラグメンテーションを解消する第4の変形例について説明する。第4の変形例では、メモリ内の領域を、アドレス値が格納されるアドレス値用メモリ領域と、数値等のアドレス値以外が格納される非アドレス値用メモリ領域とに分割し、メモリ内のポインタ変数に対して、モジュールを識別するための識別番号と、モジュールのデータ領域へのアドレスを設定するか、コード領域へのアドレスを設定するかを識別する識別フラグ（識別子）とを設け、モジュールの実行時において、ポインタ変数にアドレスを代入する際に、識別番号と識別フラグとを設定し、コンパクション後、アドレス値用メモリ領域のみについて、識別番号及び識別フラグに基づいてアドレス値を再計算するものである。

【0336】

図26は、第4の変形例におけるフラグメンテーションを解消するためのコンパイル装置及び端末装置の構成を示すブロック図である。なお、他の実施の形態と同じ構成物については、同じ符号を付し、説明を省略する。

【0337】

図26に示すように、サーバ11は、通信部22及びコンパイル装置14を備えて構成される。コンパイル装置14は、プログラム実行部16及びプログラム記憶部17を備えて構成される。プログラム実行部16は、識別番号及び識別フラグ設定プログラム生成部23j、コンパクションプログラム生成部23b及び再計算プログラム生成部23kを備えて構成される。

【0338】

識別番号及び識別フラグ設定プログラム生成部23jは、プログラムモジュールの実行時において、ポインタにアドレス値を代入する際に、プログラムモジュールを識別するための識別番号と、プログラムモジュールのデータ領域へのアドレスを設定するか、コード領域へのアドレスを設定するかを識別する識別フラグとを設定するための設定プログラムを生成する。

【0339】

再計算プログラム生成部23kは、メモリ19がコンパクションされた後、識別番号及び識別子に基づいてアドレス値を再計算するための再計算プログラムを生成する。

【0340】

端末装置12は、プログラム実行部18、メモリ19及び通信部21を備えて構成される。プログラム実行部18は、識別番号及び識別フラグ設定部24j、コンパクション部24b及び再計算部24kを備えて構成される。

【0341】

メモリ19は、アドレス値を格納するアドレス値用メモリ領域と、数値等のアドレス値以外を格納する非アドレス値用メモリ領域に分割されている。また、メモリ19には、プログラムモジュールを識別するための識別番号と、プログラムモジュールのデータ領域へのアドレスを設定するか、コード領域へのアドレスを設定するかを識別する識別フラグとを格納する領域が設けられている。

【0342】

識別番号及び識別フラグ設定部24jは、プログラムモジュールの実行時において、ポインタにアドレス値を代入する際に、プログラムモジュールを識別するための識別番号と、プログラムモジュールのデータ領域へのアドレスを設定するか、コード領域へのアドレスを設定するかを識別する識別フラグとを設定する。

【0343】

再計算部24kは、メモリ19がコンパクションされた後、識別番号及び識別子に基づいてアドレス値を再計算する。

【0344】

まず、本変形例では、プログラムモジュールがメモリに記憶される際に、当該プログラムモジュールを識別する識別番号が付与され、この識別番号と当該プログラムモジュールのデータ領域の先頭アドレス及びコード領域の先頭アドレスとが対応付けられたテーブルが予め記憶される。そして、識別番号及び識別フラグ設定部 24 j によって、プログラムモジュールの実行時において、ポインタにアドレス値を代入する際に、プログラムモジュールを識別するための識別番号と、プログラムモジュールのデータ領域へのアドレスを設定するか、コード領域へのアドレスを設定するかを識別する識別フラグとが設定される。なお、ポインタに対して、プログラムモジュールのデータ領域へのアドレスを設定する場合、識別フラグには 1 が設定され、プログラムモジュールのコード領域へのアドレスを設定する場合、識別フラグには 0 が設定される。そして、コンパクション部 24 b によって、コンパクションが行われる。

【0345】

メモリ 19 がコンパクションされた後、再計算部 24 k によって、識別番号及び識別子に基づいて、予め記憶されているテーブルを参照し、アドレス値が再計算され、再計算されたアドレス値が新たなアドレス値として設定される。

【0346】

なお、本変形例では、メモリ 19 を、アドレス値を格納するアドレス値用メモリ領域と、アドレス値以外を格納する非アドレス値用メモリ領域とに分割するとしているが、本発明は特にこれに限定されず、アドレス値の場所を示す情報を保持してもよい。

【0347】

このように、メモリ 19 内において、モジュールを識別するための識別番号と、モジュールのデータ領域へのアドレスを設定するか、コード領域へのアドレスを設定するかを識別する識別フラグとを設け、識別番号及び識別フラグ設定プログラム生成部 23 j によって、ポインタにアドレス値を代入する際に識別番号と識別フラグとを設定するための設定プログラムが生成される。コンパクションプログラム生成部 23 b によって、メモリ 19 をコンパクションするためのコンパクションプログラムが生成される。メモリ 19 がコンパクションされた後、再計算プログラム生成部 23 k によって、識別番号及び識別フラグに基づいてアドレス値を再計算するための再計算プログラムが生成される。したがって、メモリのフラグメンテーションを解消することができ、メモリを有効的に利用することができる。

【0348】

次に、メモリのフラグメンテーションを解消する第 5 の変形例について説明する。第 5 の変形例では、メモリ内の領域を、アドレス値が格納されるアドレス値用メモリ領域と、数値等のアドレス値以外が格納される非アドレス値用メモリ領域とに分割し、メモリ内のポインタ変数に対して、モジュールを識別するための識別番号と、モジュールのデータ領域へのアドレスを設定するか、コード領域へのアドレスを設定するかを識別する識別フラグ（識別子）とを設け、モジュールの実行時において、ポインタ変数にアドレス値を代入する際に、識別番号と識別フラグとを設定し、当該アドレス値には、データ領域の先頭アドレスからのオフセット値又はコード領域の先頭アドレスからのオフセット値を設定する。なお、プログラムモジュールの実行時に、ポインタ変数を使用する際は、データ領域の先頭アドレス又はコード領域の先頭アドレスにオフセット値を加算することによって実アドレスに変換して使用する。

【0349】

なお、本変形例と上述の第 4 の変形例とは、アドレス値をオフセット値として記憶する点のみが異なるため、説明を省略する。

【0350】

このように、メモリ 19 内において、モジュールを識別するための識別番号と、モジュールのデータ領域へのアドレスを設定するか、コード領域へのアドレスを設定するかを識別する識別フラグとを設け、ポインタにアドレス値を代入する際に識別番号と識別フラグとを設定するとともに、データ領域の先頭アドレス又はコード領域の先頭アドレスからの

オフセット値を設定するためのオフセット値設定プログラムが生成される。そして、ポインタを使用する際に、オフセット値にデータ領域の先頭アドレス又はコード領域の先頭アドレスを加算することによって実アドレスに変換するための実アドレス変換プログラムが生成される。したがって、メモリのフラグメンテーションを解消することができ、メモリを有効的に利用することができる。

【0351】

次に、メモリのフラグメンテーションを解消する第6の変形例について説明する。第6の変形例では、リファレンスがオブジェクトを間接的に参照するハンドルを設け、コンパクションによって移動するコード領域又はデータ領域を指すハンドルがあるか否かを検索し、コード領域又はデータ領域を指すハンドルが発見された場合、当該ハンドルを再配置し、全てのハンドルについて再配置を行った後、当該コード領域又はデータ領域をコンパクションするものである。

【0352】

図27は、第6の変形例におけるフラグメンテーションを解消するためのコンパイル装置及び端末装置の構成を示すブロック図である。なお、他の実施の形態と同じ構成物については、同じ符号を付し、説明を省略する。

【0353】

図27に示すように、サーバ11は、通信部22及びコンパイル装置14を備えて構成される。コンパイル装置14は、プログラム実行部16及びプログラム記憶部17を備えて構成される。プログラム実行部16は、ハンドル検索プログラム生成部231、コンパクションプログラム生成部23b及びハンドル再配置プログラム生成部23mを備えて構成される。

【0354】

ハンドル検索プログラム生成部231は、コンパクションによって移動するプログラムモジュールのコード領域又はデータ領域を指すハンドルがあるか否かを検索するための検索プログラムを生成する。

【0355】

コンパクションプログラム生成部23bは、プログラムモジュールのコード領域又はデータ領域毎にコンパクションするためのコンパクションプログラムを生成する。

【0356】

ハンドル再配置プログラム生成部23mは、コード領域又はデータ領域を指すハンドルが発見された場合、当該ハンドルを再配置するためのハンドル再配置プログラムを生成する。

【0357】

また、コンパクションプログラム生成部23bは、全てのハンドルについて再配置を行った後、当該コード領域又はデータ領域をコンパクションするためのコンパクションプログラムを生成する。

【0358】

端末装置12は、プログラム実行部18、メモリ19及び通信部21を備えて構成される。プログラム実行部18は、ハンドル検索部241、コンパクション部24b及びハンドル再配置部24mを備えて構成される。

【0359】

メモリ19には、リファレンスがオブジェクトのアドレスを間接的に参照するためのハンドルのための領域が設けられている。したがって、リファレンスは、オブジェクトを直接的に参照するのではなく、ハンドルを介して間接的に参照する。ハンドルとオブジェクトとの関係は一つ一つに対応している。これは、オブジェクトを指すハンドルが複数存在しないことを意味している。また、リファレンスとオブジェクトとの関係は多対一に対応している。オブジェクトは、複数の異なるリファレンスから参照される可能性があるためである。

【0360】

ハンドル検索部 241 は、コンパクションによって移動するプログラムモジュールのコード領域又はデータ領域を指すハンドルがあるか否かを検索する。コンパクション部 24b は、プログラムモジュールのコード領域又はデータ領域毎にコンパクションする。ハンドル再配置部 24m は、コード領域又はデータ領域を指すハンドルが発見された場合、当該ハンドルのアドレス値を再配置する。また、コンパクション部 24b は、全てのハンドルについて再配置を行った後、当該コード領域又はデータ領域をコンパクションする。

【0361】

このように、ハンドル検索プログラム生成部 231 によって、コンパクションによって移動するコード領域又はデータ領域を指すハンドルがあるか否かを検索するための検索プログラムが生成される。そして、コード領域又はデータ領域を指すハンドルが発見された場合、ハンドル再配置プログラム生成部 23m によって、当該ハンドルを再配置するためのハンドル再配置プログラムが生成される。全てのハンドルについて再配置を行った後、コンパクションプログラム生成部 23b によって、当該コード領域又はデータ領域をコンパクションするためのコンパクションプログラムが生成される。したがって、メモリのフラグメンテーションを解消することができ、メモリを有効的に利用することができる。

【0362】

また以上説明したような原始プログラム、目的プログラム、その他のプログラムの全てはコンピュータ読み取り可能な記録媒体に格納されて、コンピュータから読み出されて利用又は実行され、場合によってはインターネット等のネットワークを介して他のコンピュータと通信される。同様に、このような翻訳を行うコンパイラプログラムもまたコンピュータ読み取り可能な記録媒体に格納されて、コンピュータから読み出されてコンパイル（原始プログラムから目的プログラムへの翻訳・変換）が実行され、場合によってはインターネット等のネットワークを介して他のコンピュータと通信される。

【0363】

このコンパイラプログラムによってコンピュータが行うのはこのコンパイラプログラムによって規定されたコンパイル方法であり、このコンピュータハードウェアとコンパイラプログラムとは全体としてコンパイラ装置を構成する。

【産業上の利用可能性】

【0364】

本発明に係るコンパイラプログラム、コンパイラプログラムを記録したコンピュータ読み取り可能な記録媒体、コンパイル方法及びコンパイル装置は、自動的に再入可能な目的プログラムを生成することができ、あるプログラム言語で記述された原始プログラムを、あるコンピュータによって実行するための目的プログラムに変換するためのコンパイラプログラム、コンパイラプログラムを記録したコンピュータ読み取り可能な記録媒体、コンパイル方法及びコンパイル装置等として有用である。

【図面の簡単な説明】

【0365】

【図1】 本発明の一実施の形態による動的モジュールリンクシステムの構成を示す図である。

【図2】 図1に示すコンパイル装置の構成を示すブロック図である。

【図3】 図2に示すコンパイル装置によるコンパイラ処理の一例を示すフローチャートである。

【図4】 本発明の第1の実施の形態におけるコンパイル装置が変換するソースプログラムと、コンパイル装置によって変換されたオブジェクトプログラムとの構造を示す図である。

【図5】 図1に示す端末装置の構成を示すブロック図である。

【図6】 図5に示す端末装置によってアプリケーションプログラムモジュールがライブラリプログラムモジュールを呼び出す処理を示すフローチャートである。

【図7】 アプリケーションプログラムモジュールからライブラリプログラムモジュールを呼び出す処理を説明するための図である。

【図 8】アプリケーションプログラムモジュール及びライブラリプログラムモジュールのソースプログラム及びオブジェクトプログラムの一例を示す図である。

【図 9】スタックの状態を説明するための図である。

【図 10】第 2 の実施の形態におけるソースプログラム及びオブジェクトプログラムの一例を示す図である。

【図 11】本発明の第 2 の実施の形態におけるコンパイル装置の構成を示すブロック図である。

【図 12】図 11 に示すコンパイル装置によるコンパイラ処理の一例を示すフローチャートである。

【図 13】本発明の第 2 の実施の形態におけるライブラリプログラムモジュールのコード領域及びデータ領域を示す図である。

【図 14】本発明の第 3 の実施の形態におけるライブラリプログラムモジュールのソースプログラム及びオブジェクトプログラムの一例を示す図である。

【図 15】本発明の第 3 の実施の形態におけるライブラリプログラムモジュールのコード領域及びデータ領域を示す図である。

【図 16】本発明の第 4 の実施の形態であるコンパイラが変換の対象とする呼び出し元プログラムであるアプリケーションプログラムモジュールと、呼び出されるプログラムであるライブラリプログラムモジュールと、その呼び出し処理時に行われる処理の一例を概念的に示す図である。

【図 17】本発明の第 5 の実施の形態におけるコンパイル装置の構成を示すブロック図である。

【図 18】本発明の第 5 の実施の形態における端末装置の構成を示すブロック図である。

【図 19】図 18 に示す端末装置による動的モジュールリンク処理の一例を示すフローチャートである。

【図 20】フラグメンテーションを解消するためのコンパイル装置及び端末装置の構成を示すブロック図である。

【図 21】第 1 の変形例におけるフラグメンテーションを解消するためのコンパイル装置及び端末装置の構成を示すブロック図である。

【図 22】第 2 の変形例におけるフラグメンテーションを解消するためのコンパイル装置及び端末装置の構成を示すブロック図である。

【図 23】第 2 の変形例におけるフラグメンテーションの解消方法について説明するための図である。

【図 24】第 3 の変形例におけるフラグメンテーションを解消するためのコンパイル装置及び端末装置の構成を示すブロック図である。

【図 25】第 3 の変形例におけるフラグメンテーションの解消方法について説明するための図である。

【図 26】第 4 の変形例におけるフラグメンテーションを解消するためのコンパイル装置及び端末装置の構成を示すブロック図である。

【図 27】第 6 の変形例におけるフラグメンテーションを解消するためのコンパイル装置及び端末装置の構成を示すブロック図である。

【符号の説明】

【0366】

10 動的モジュールリンクシステム

11 サーバ

12 端末装置

13 インターネット

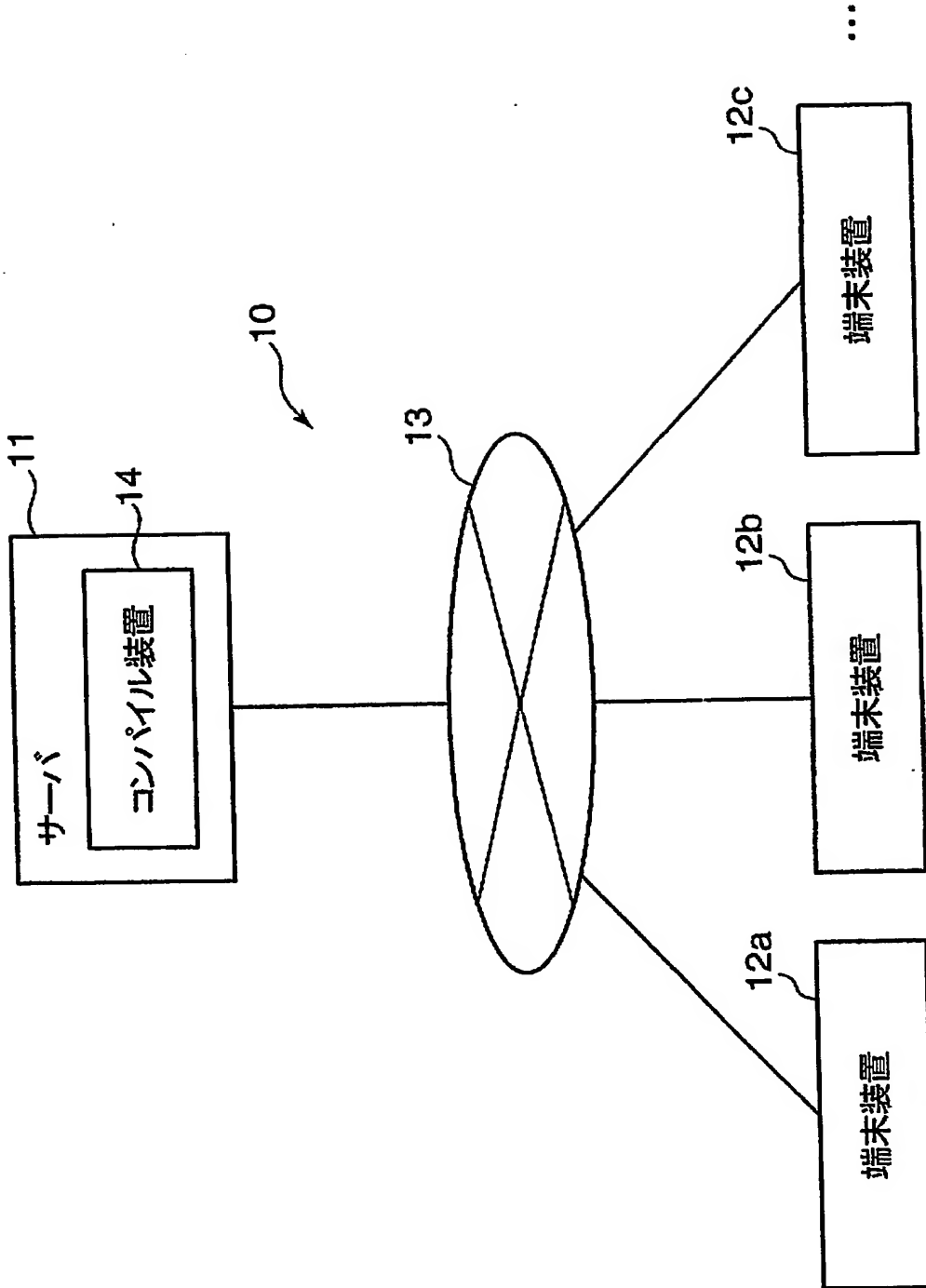
14 コンパイル装置

16, 18 プログラム実行部

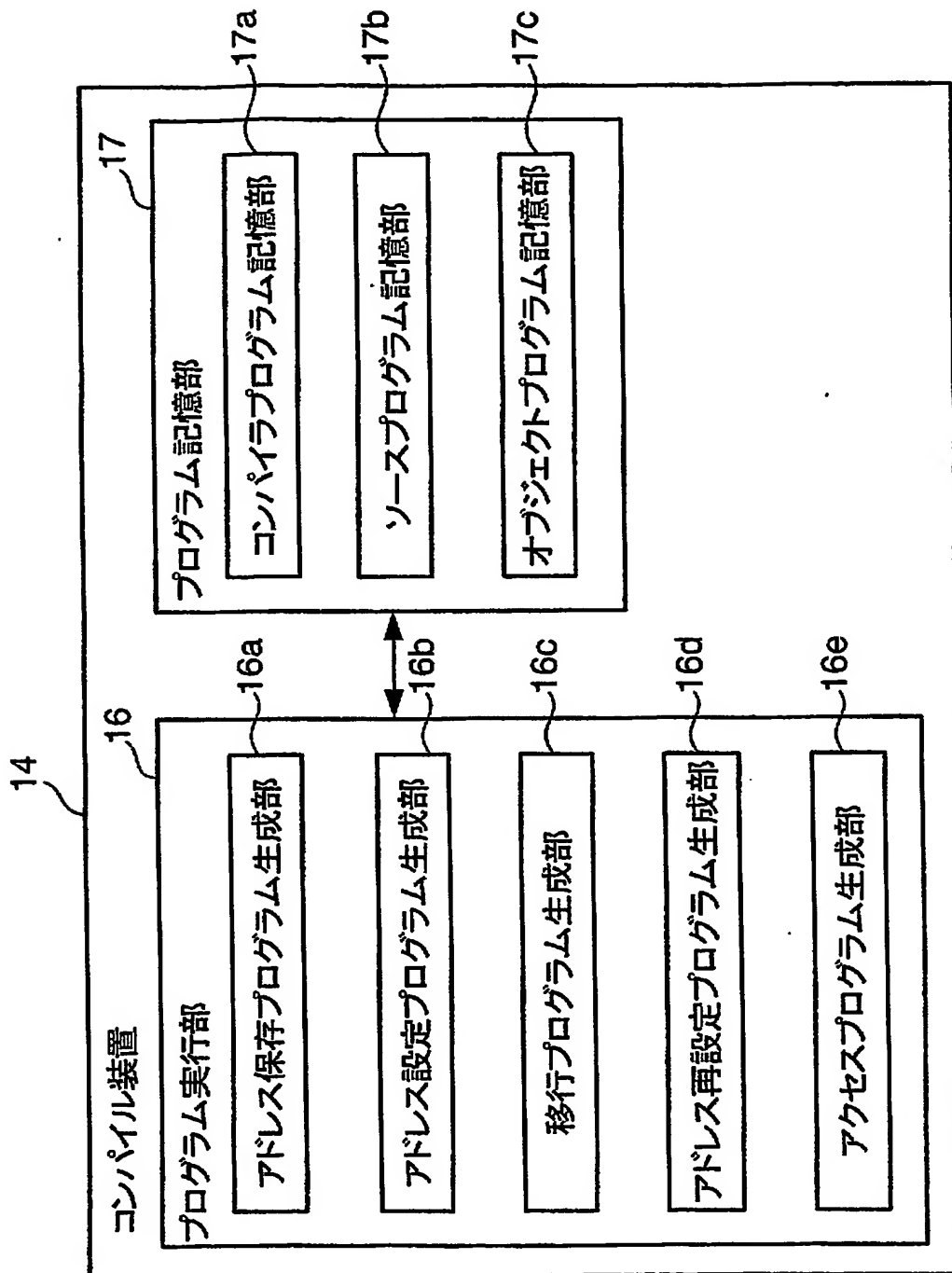
16a アドレス保存プログラム生成部

- 16b アドレス設定プログラム生成部
- 16c 移行プログラム生成部
- 16d アドレス再設定プログラム生成部
- 16e アクセスプログラム生成部
- 17 プログラム記憶部
 - 17a コンパイラプログラム記憶部
 - 17b ソースプログラム記憶部
 - 17c オブジェクトプログラム記憶部
- 18a アドレス保存部
- 18b アドレス設定部
- 18c 移行部
- 18d アドレス再設定部
- 18e アクセス部
- 19 メモリ
 - 19a アプリケーションプログラムモジュール記憶部
 - 19b ライブラリプログラムモジュール記憶部
 - 19c スタック

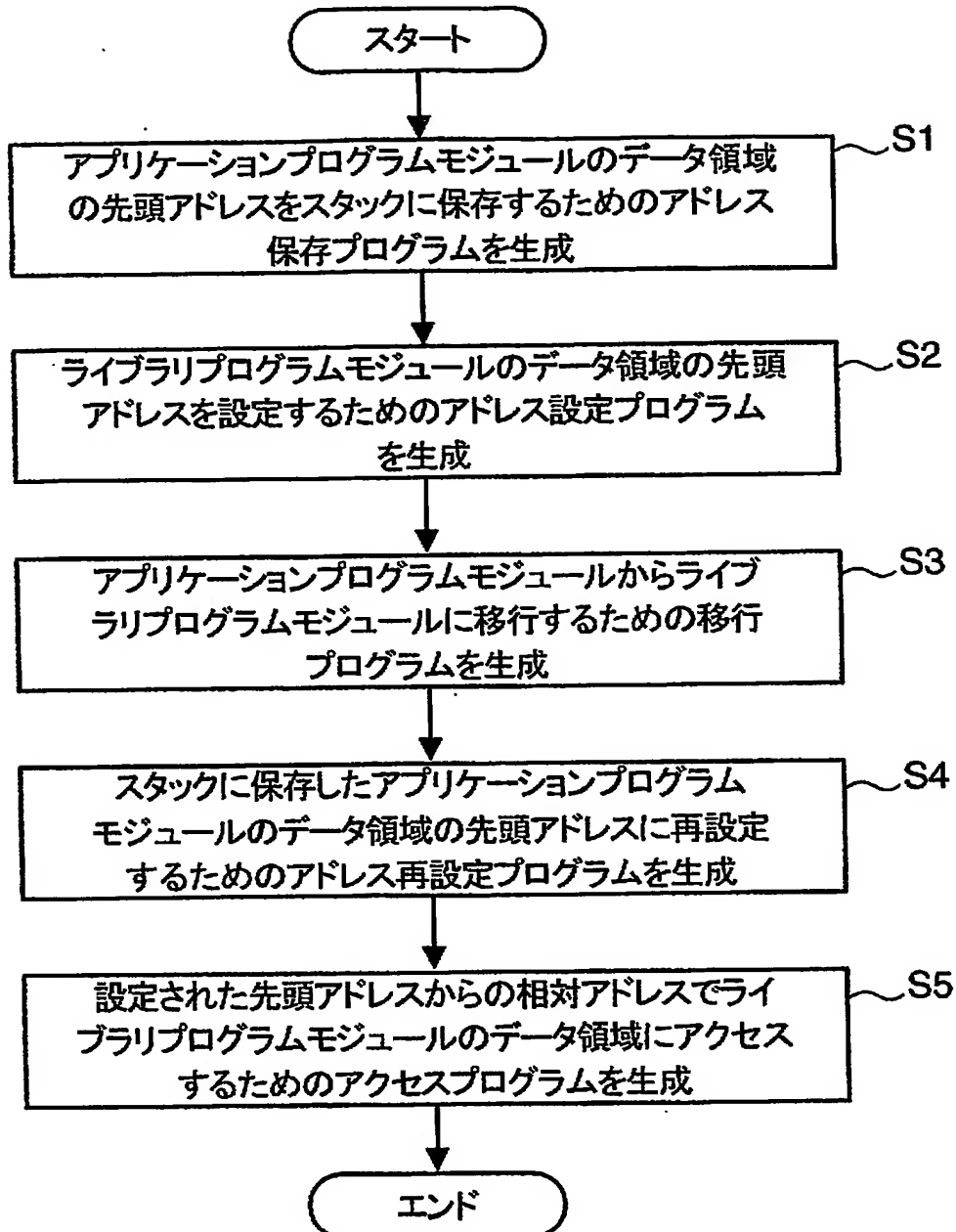
【書類名】 図面
【図 1】



【図2】

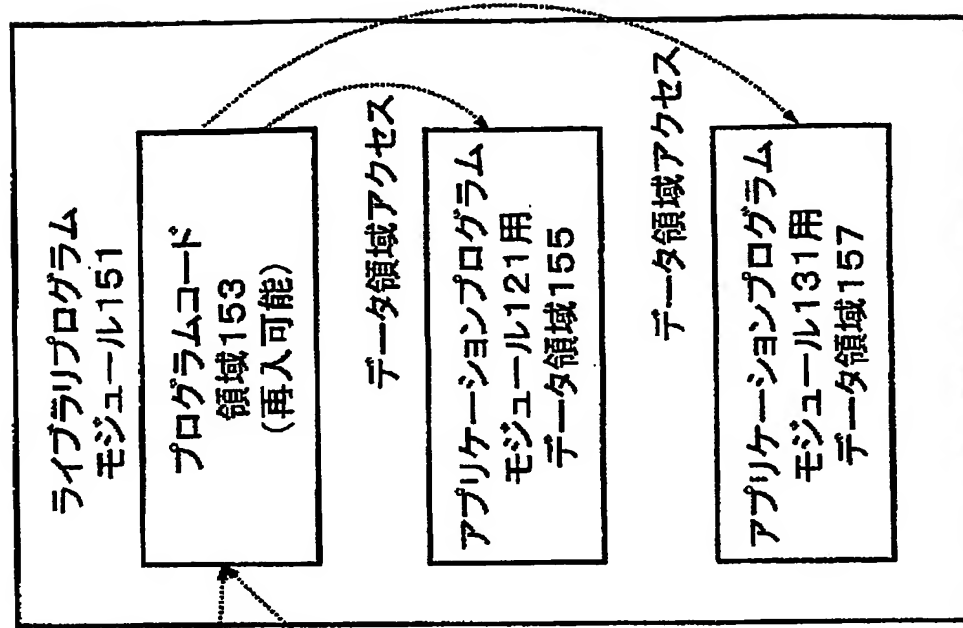


【図 3】

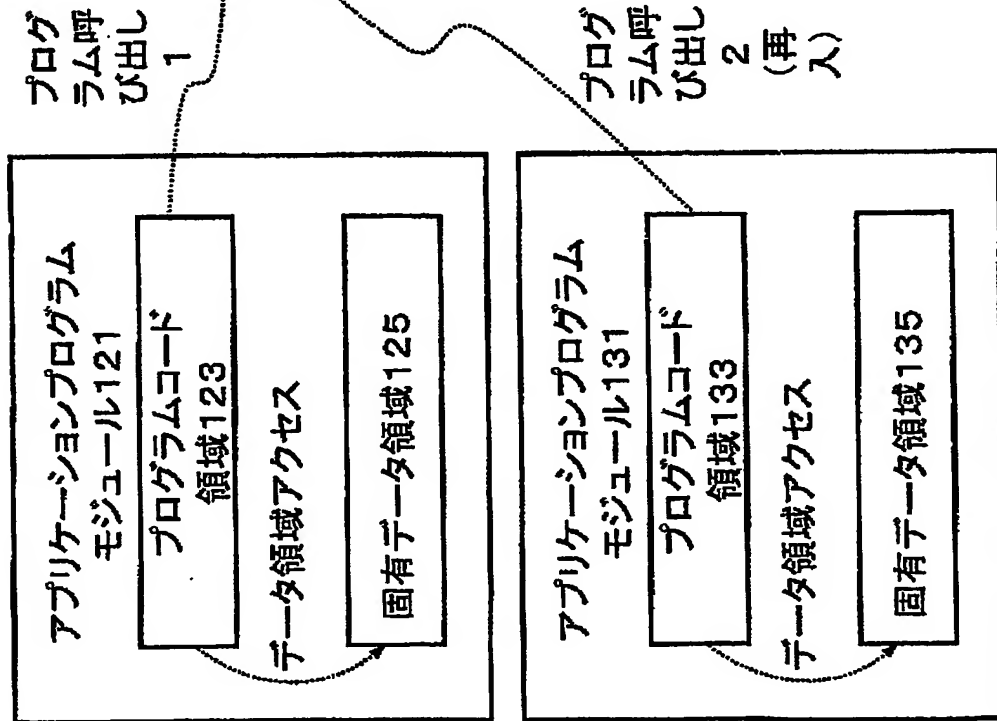


【図4】

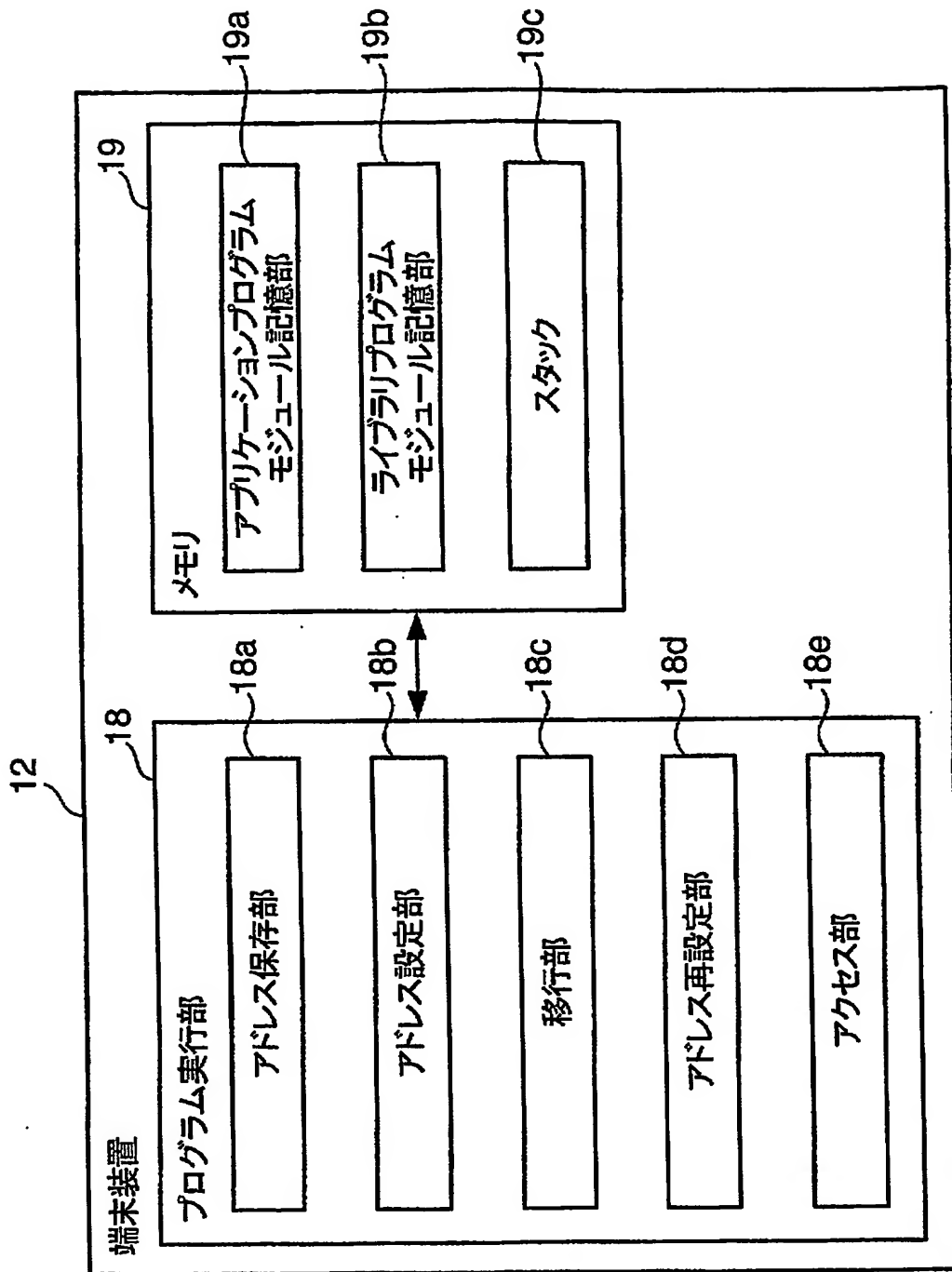
(b)



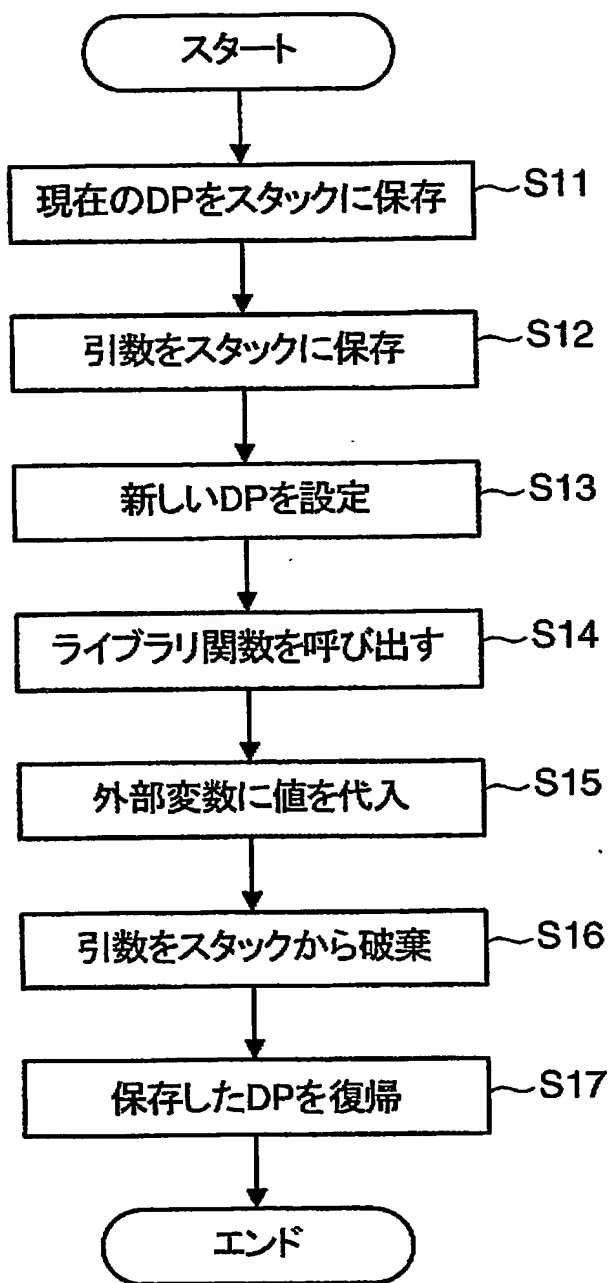
(a)



【図 5】

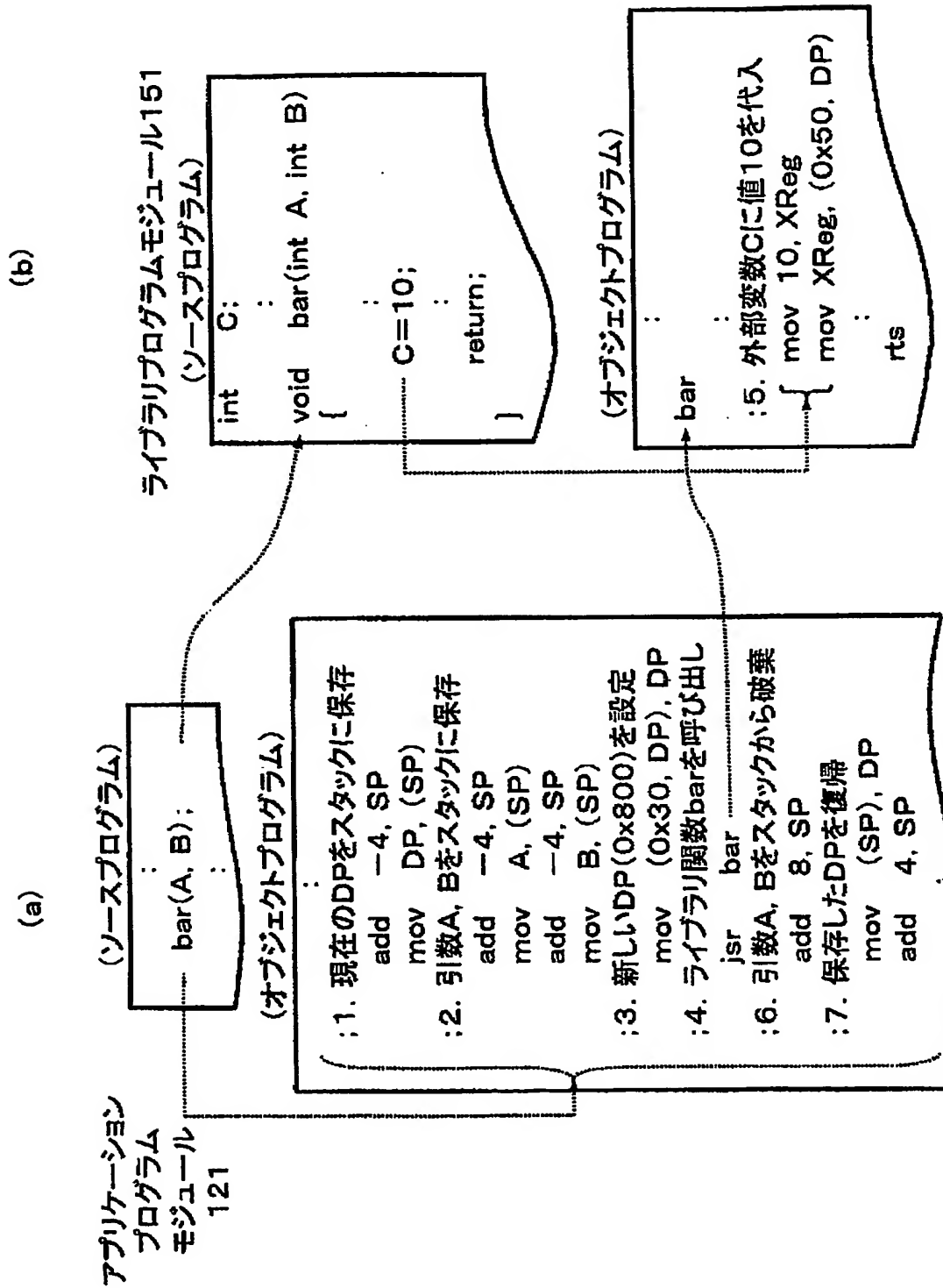


【図 6】

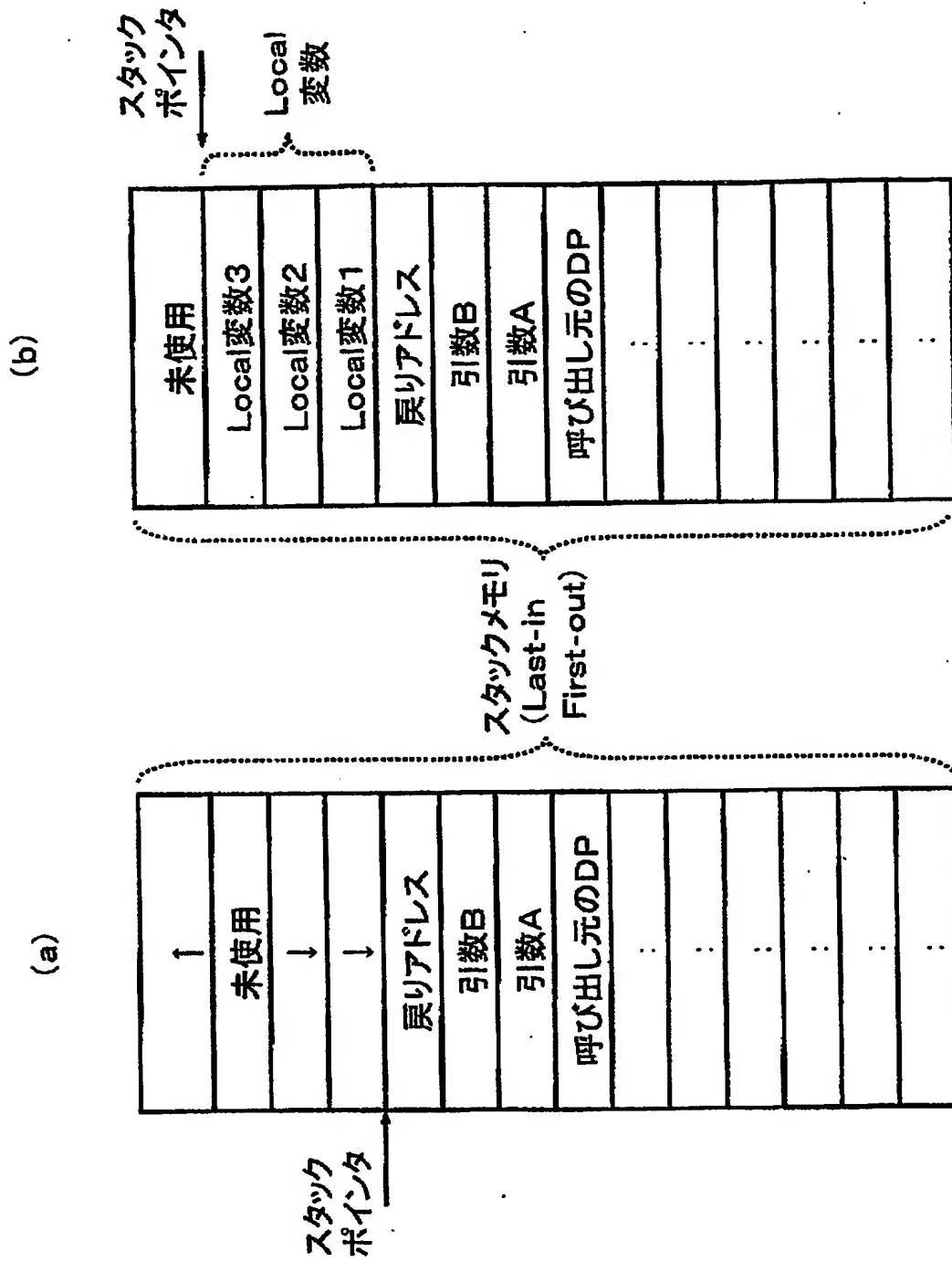




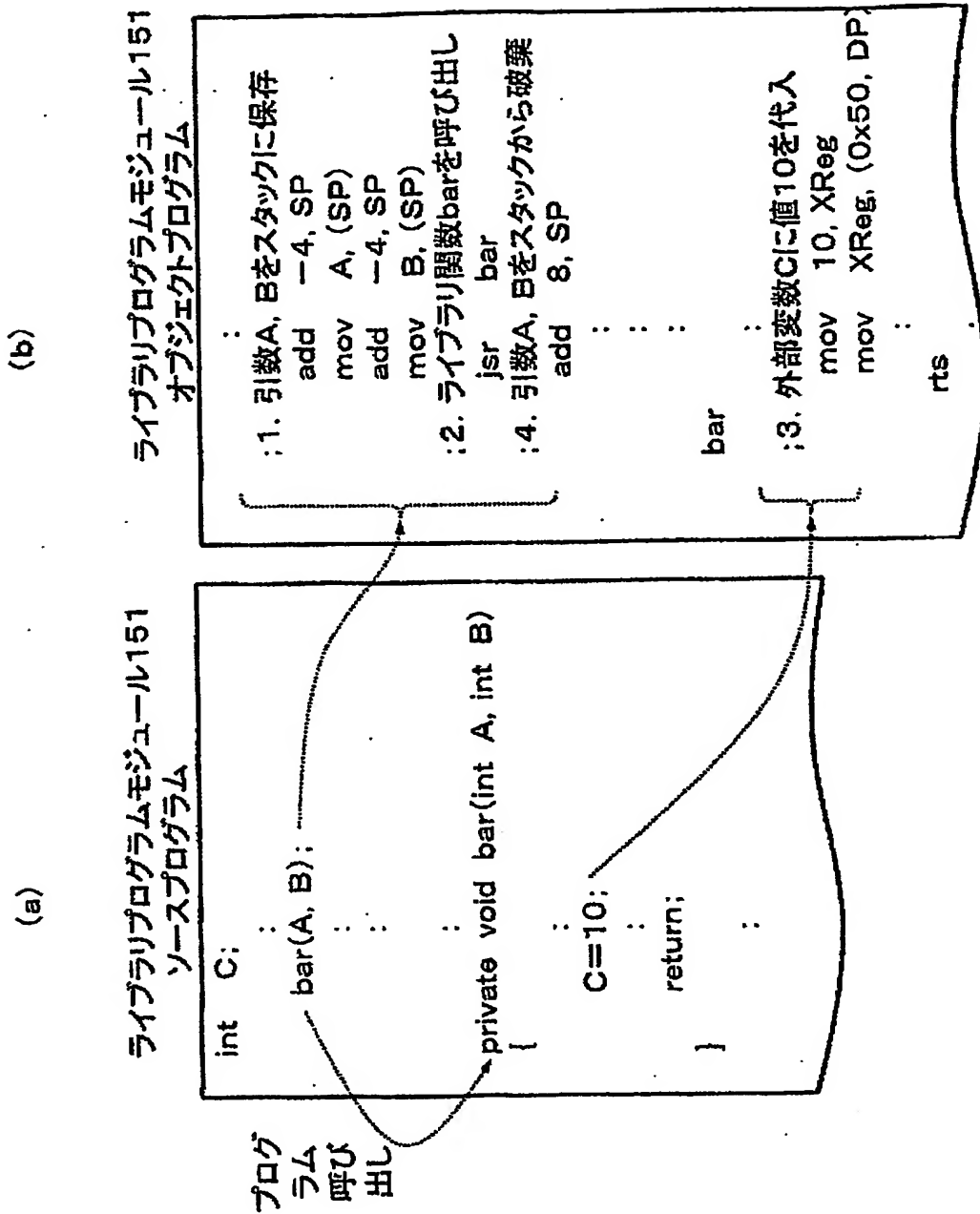
【図 8】



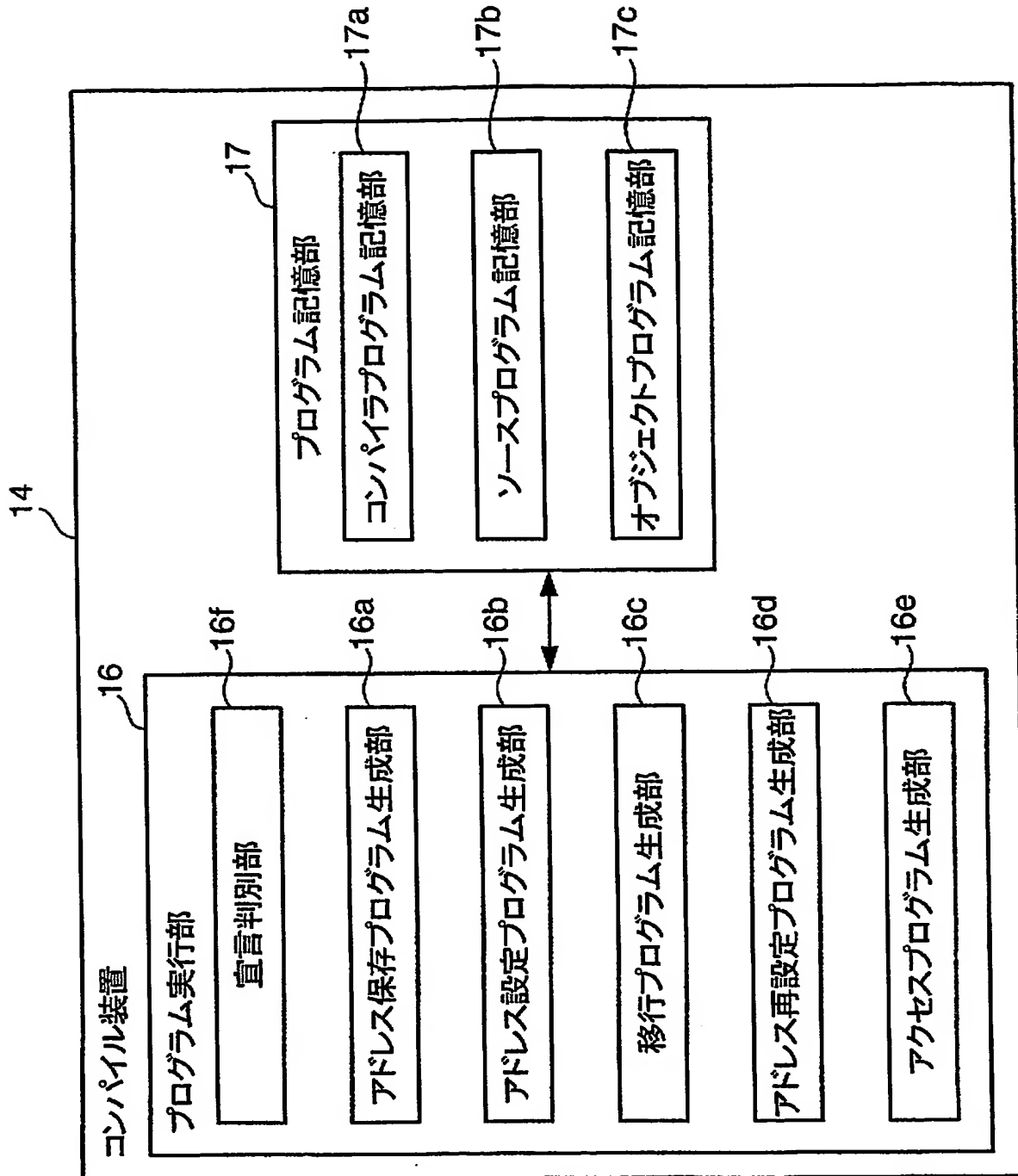
【図 9】



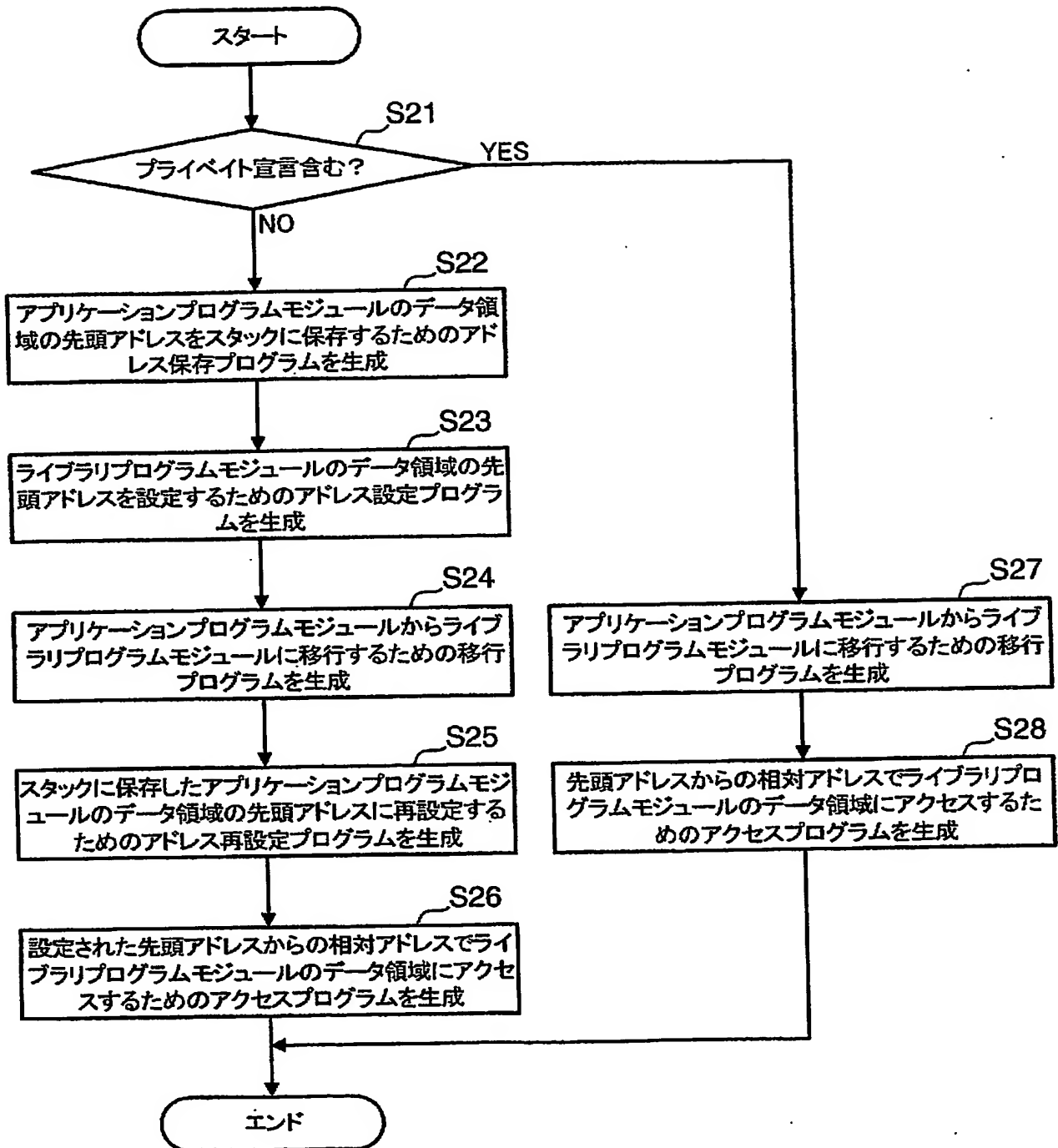
【図10】



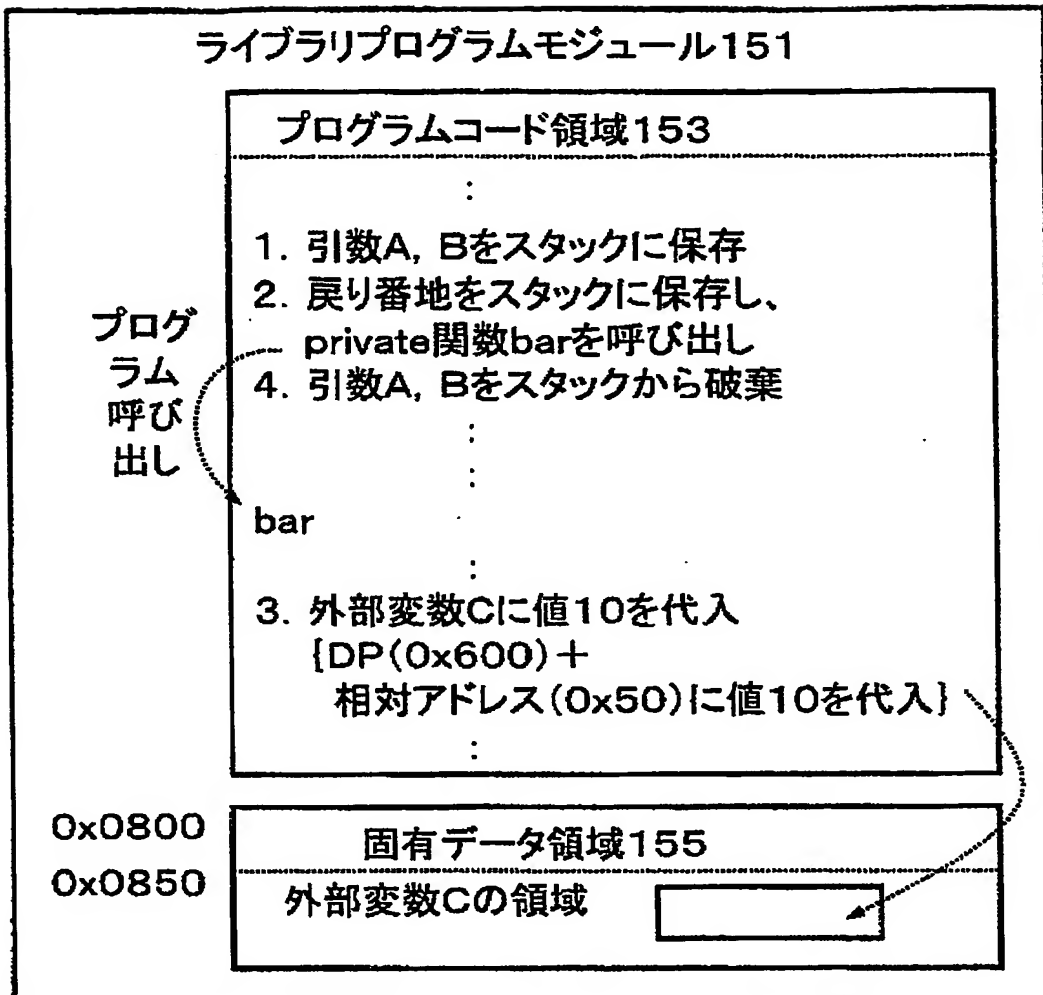
【図 11】



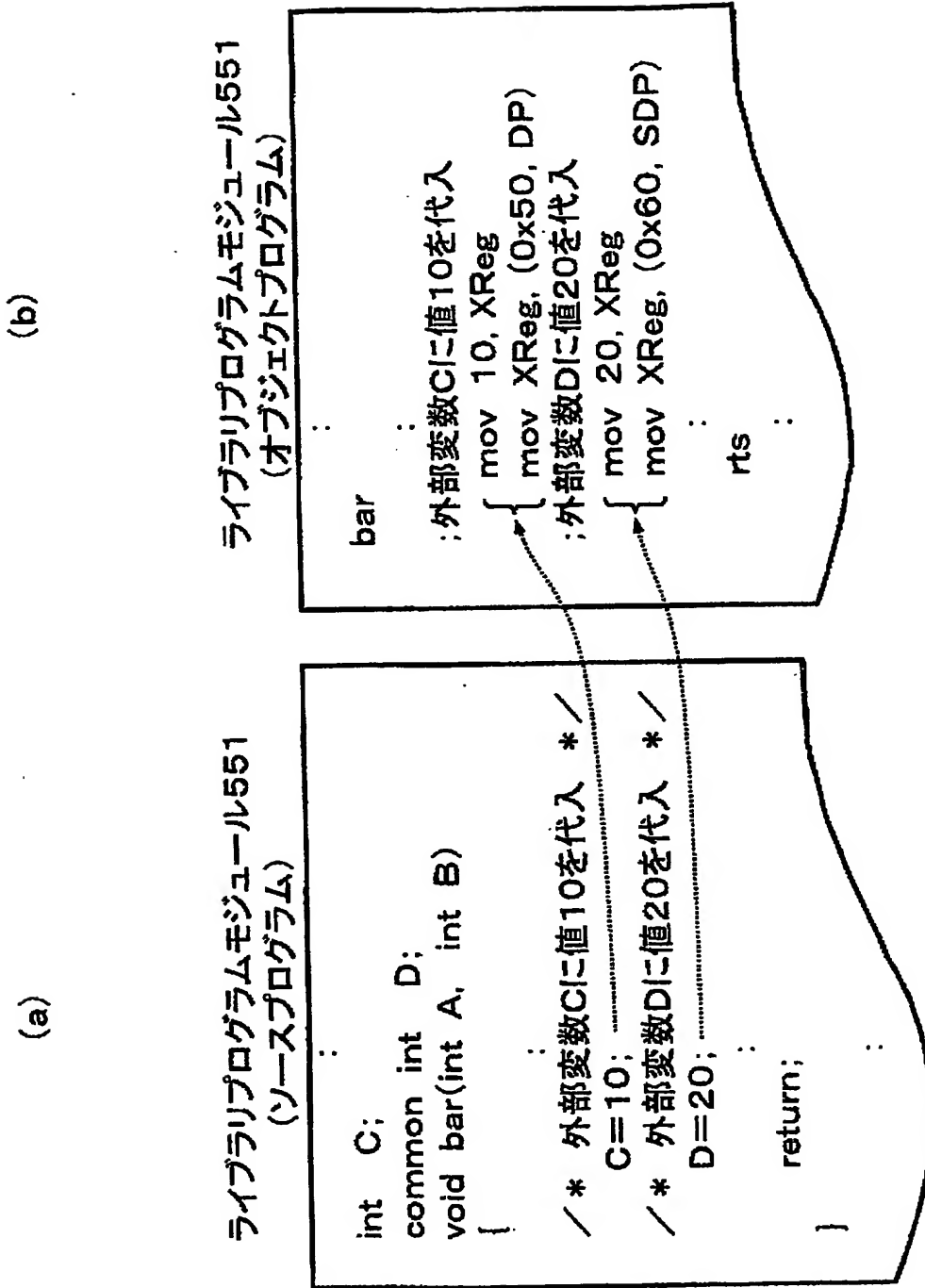
【図 12】



【図13】

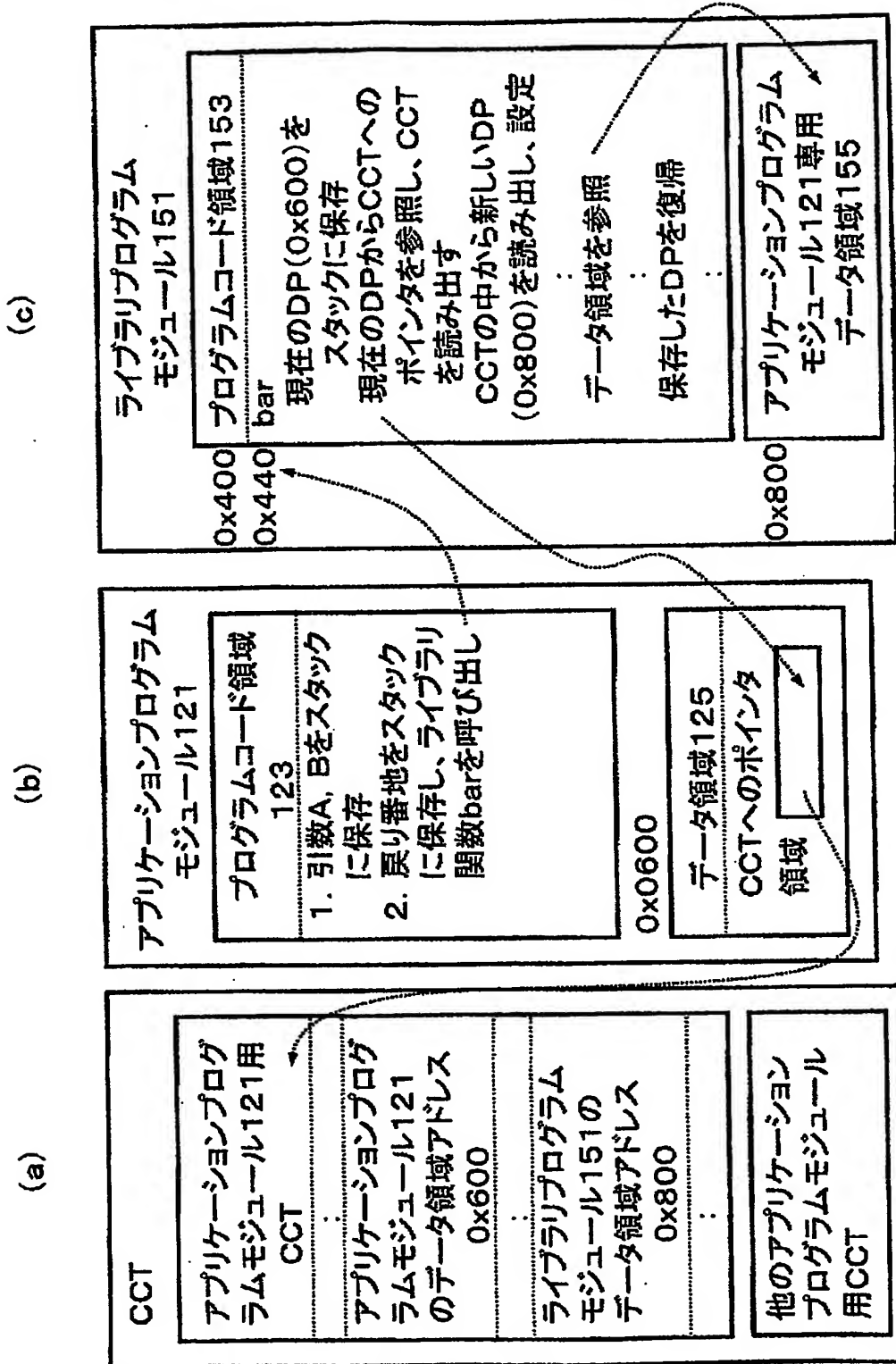


【図 14】

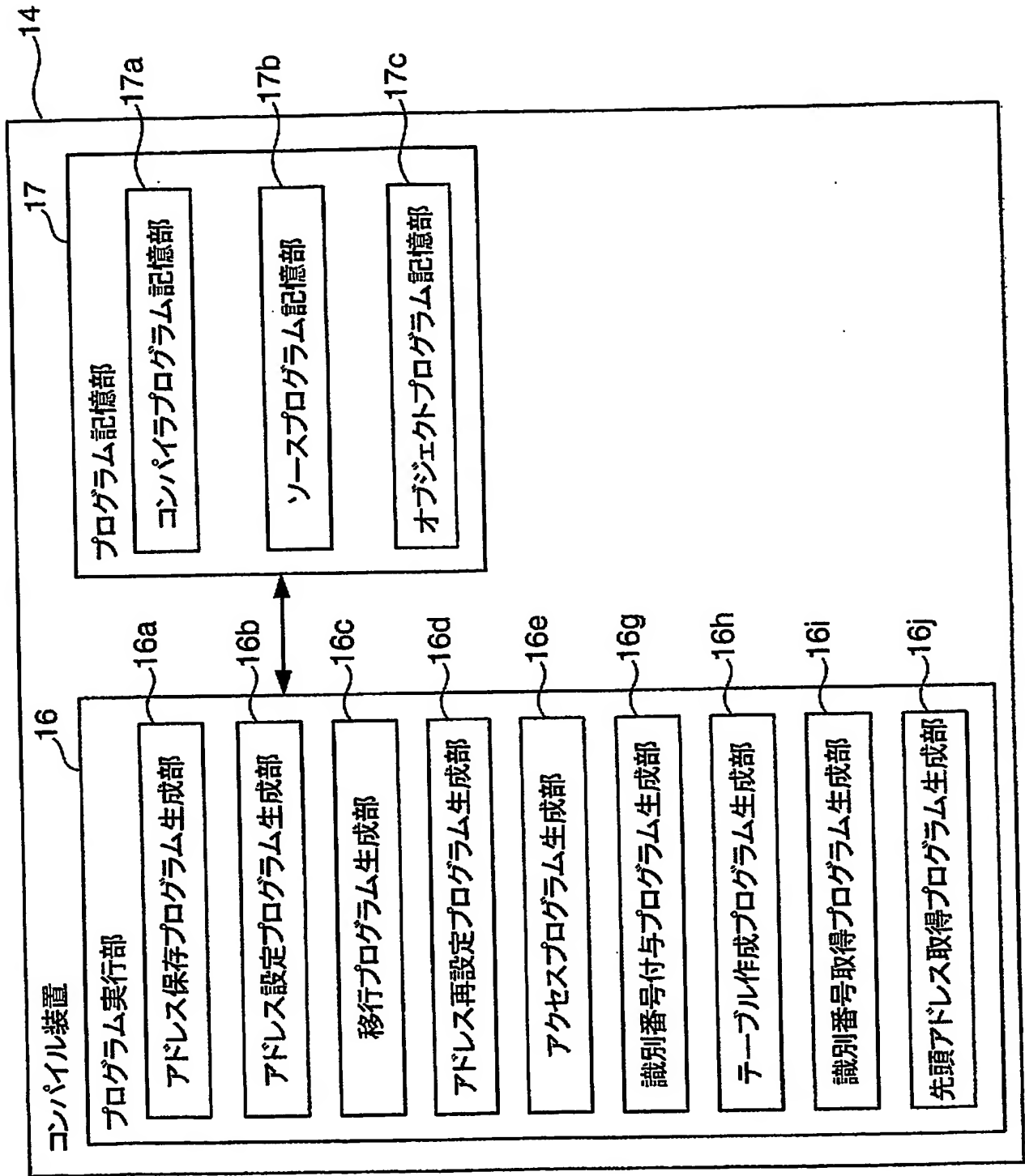




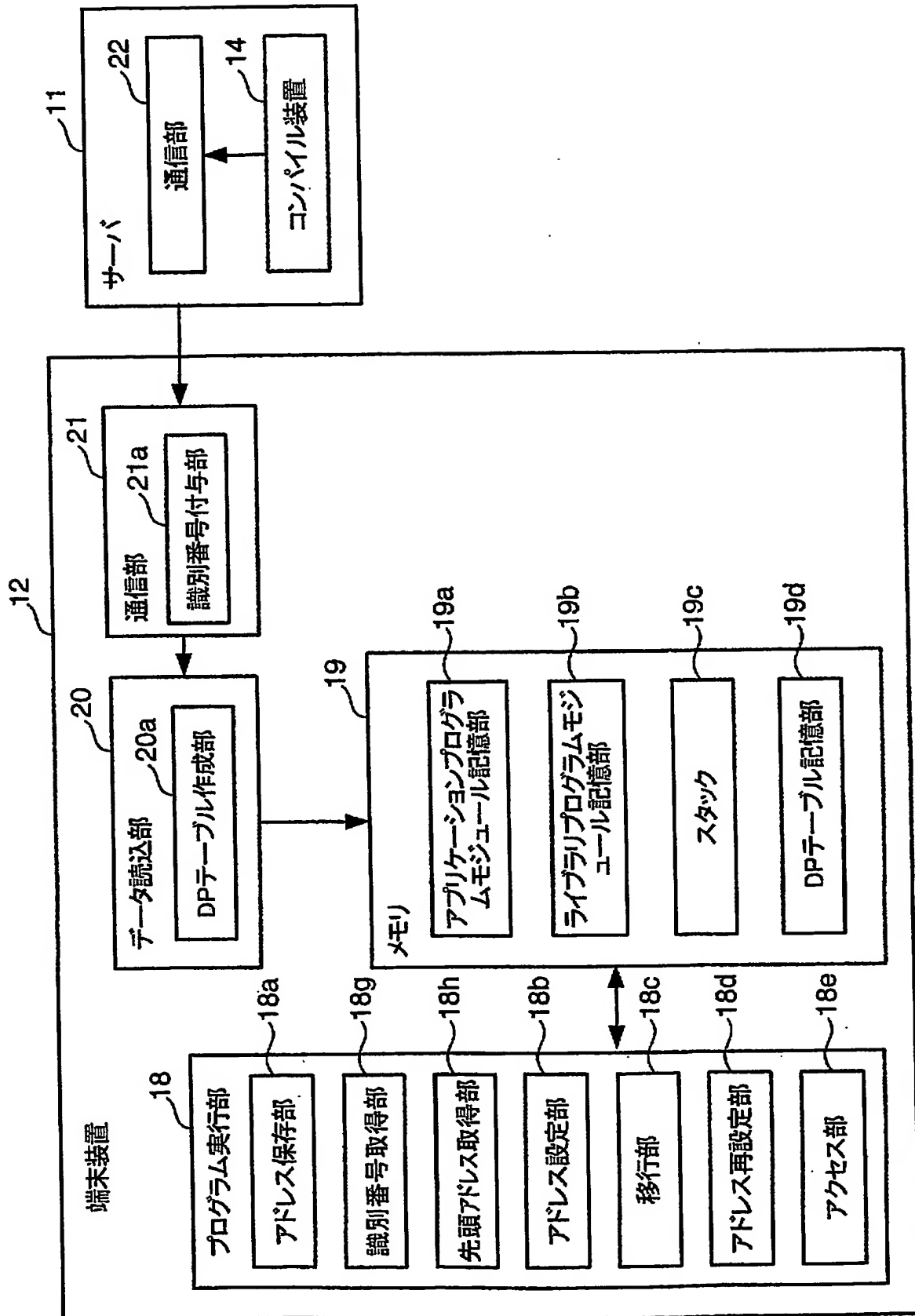
【図16】



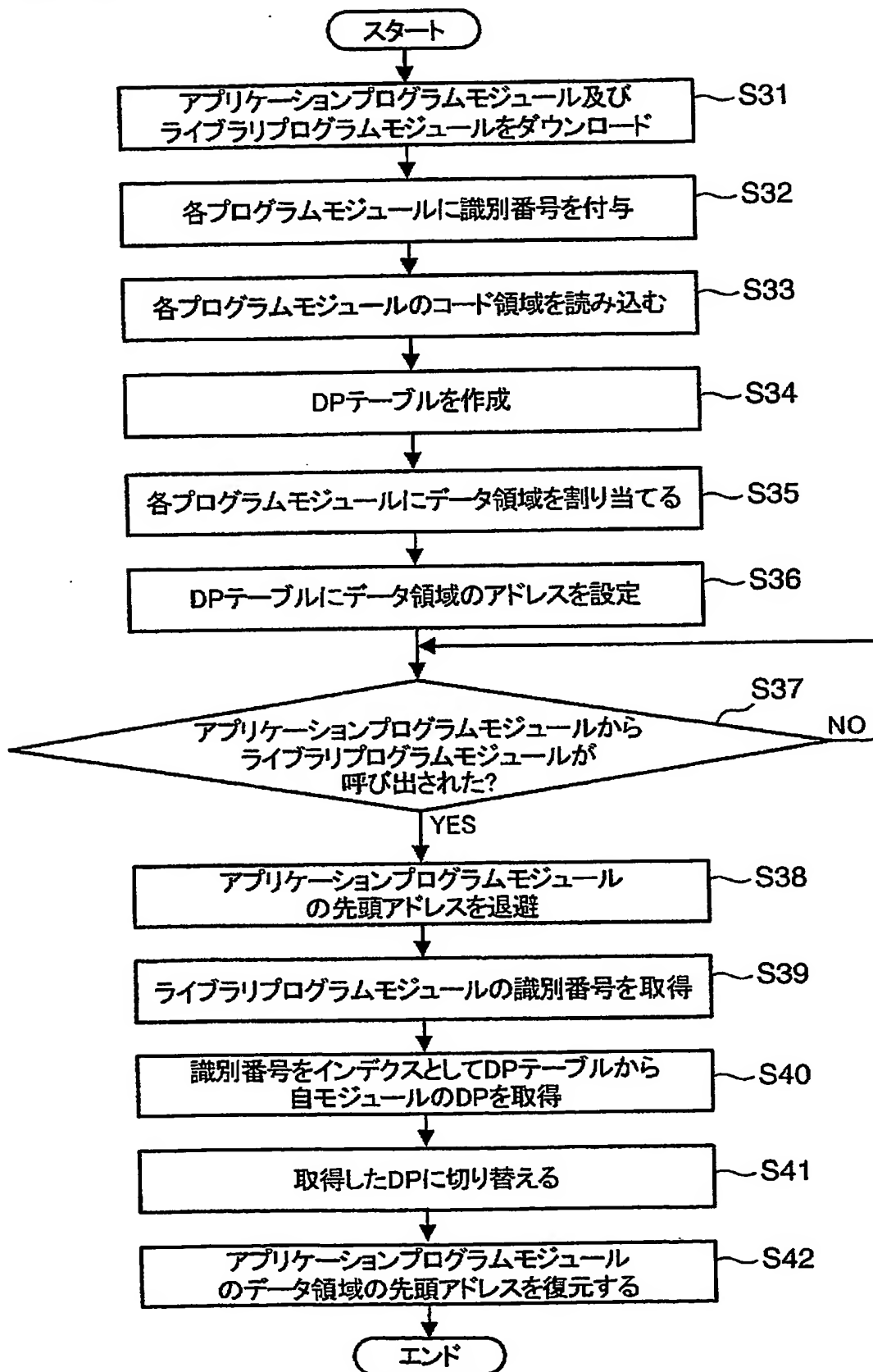
【図 17】



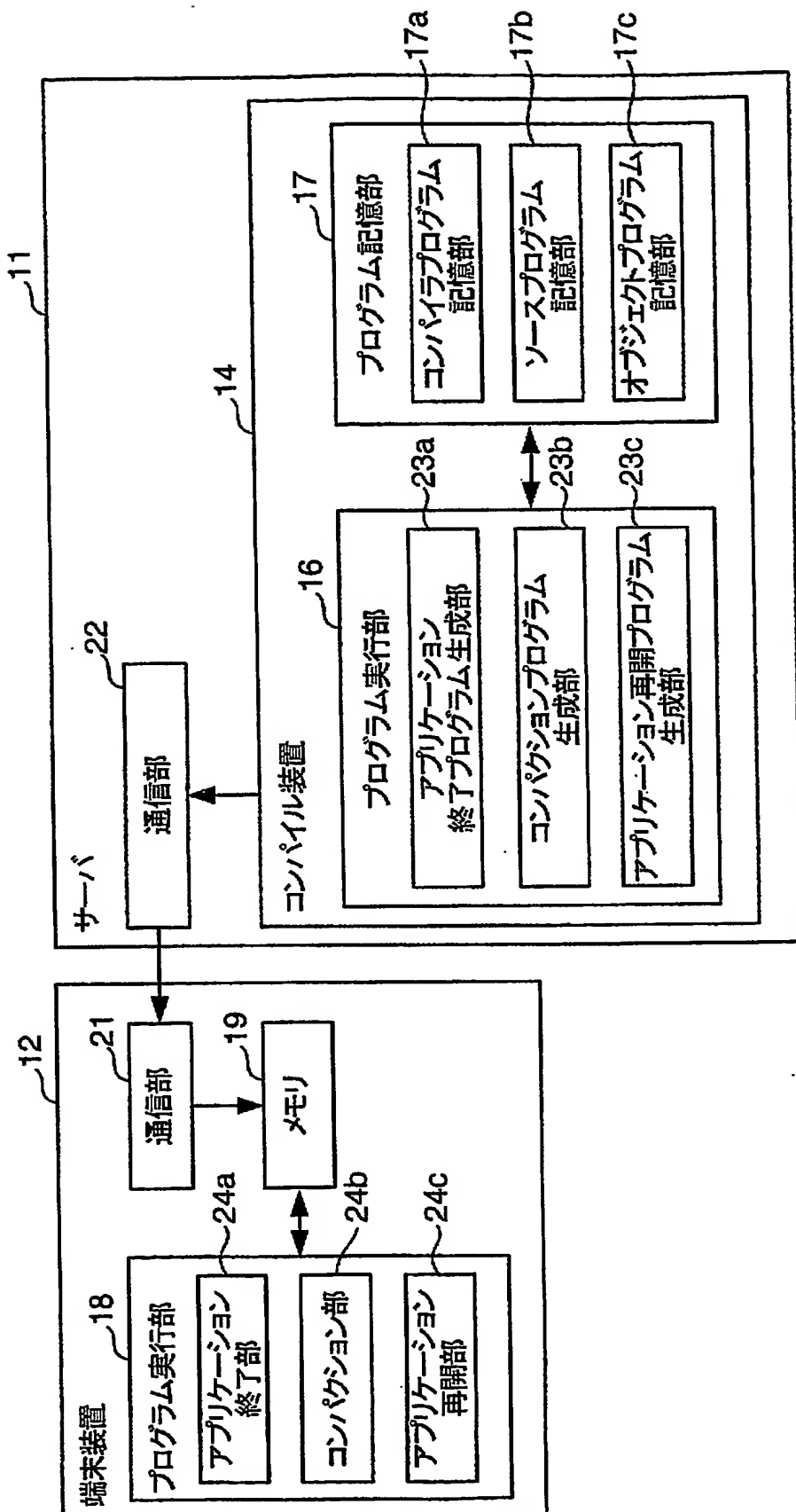
【図18】



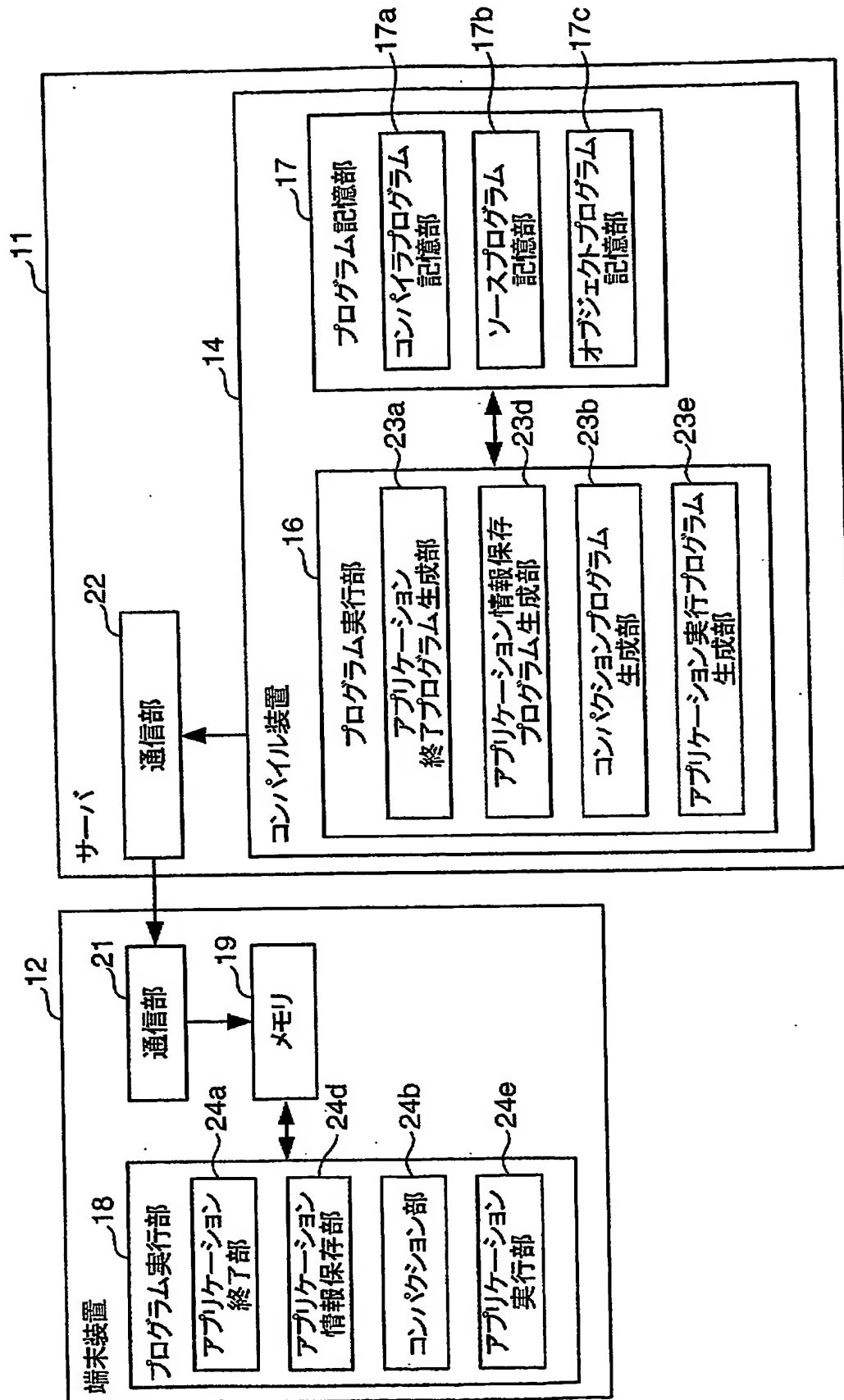
【図 19】



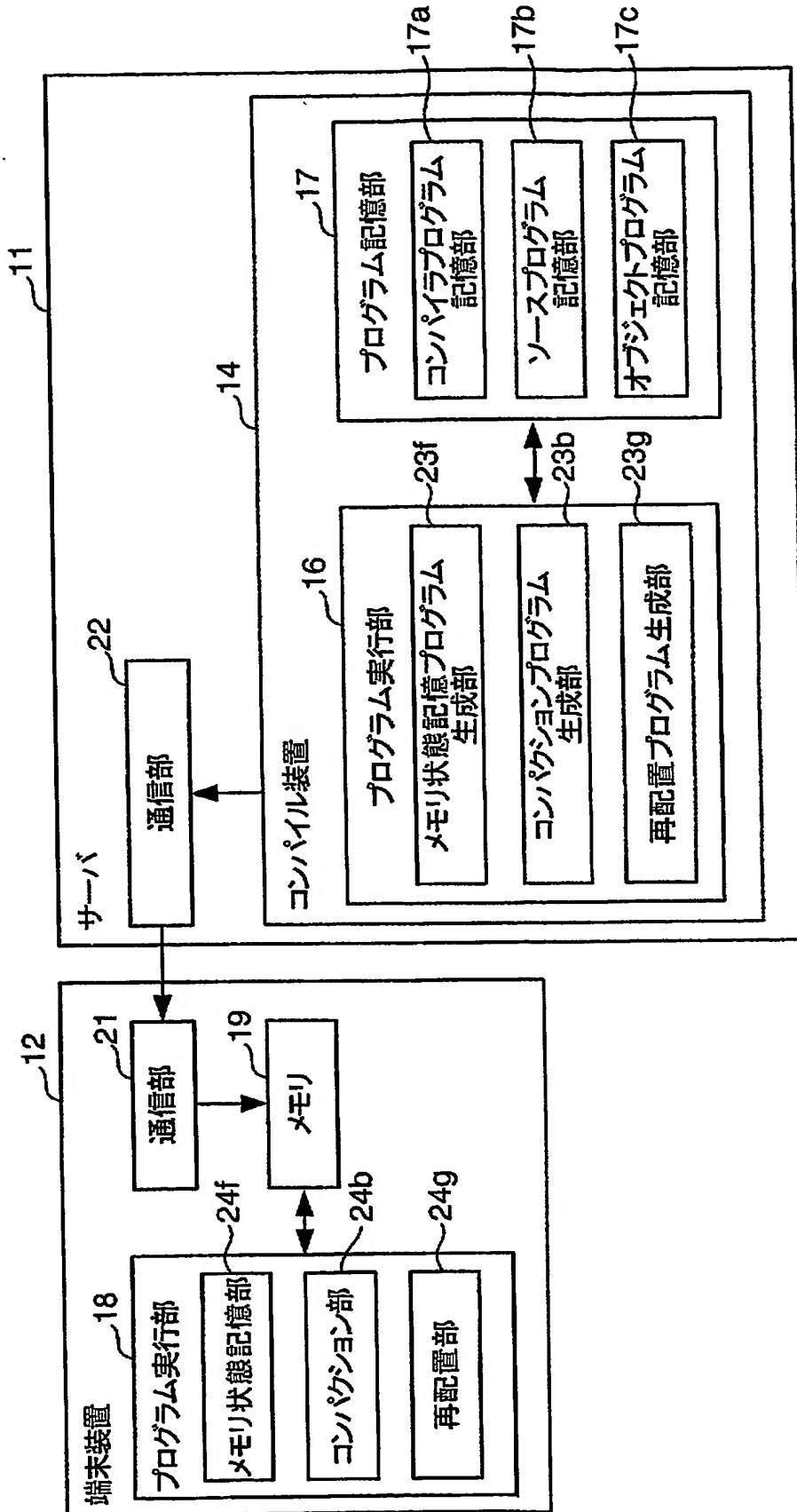
【図 20】



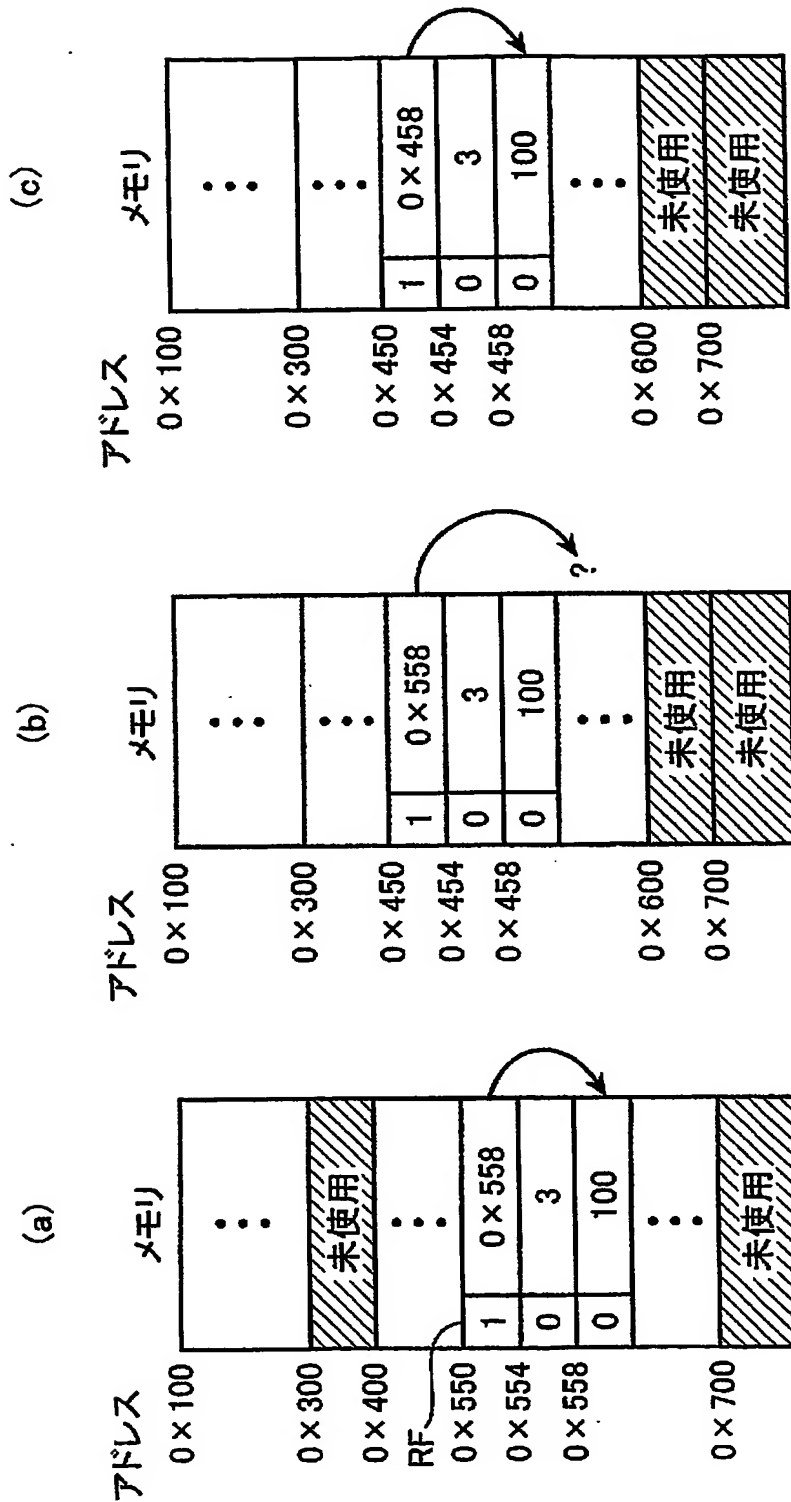
【図 21】



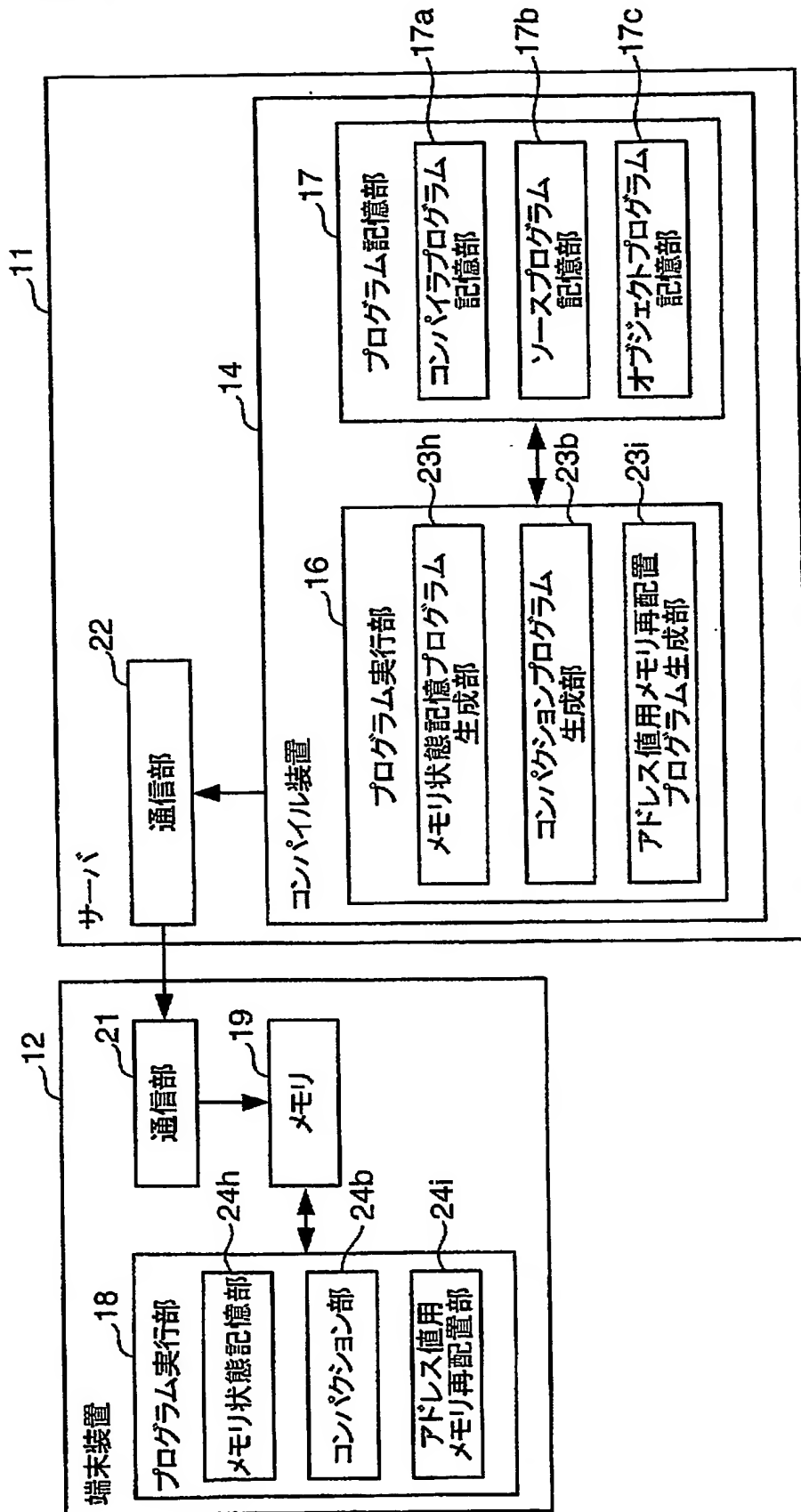
【図22】



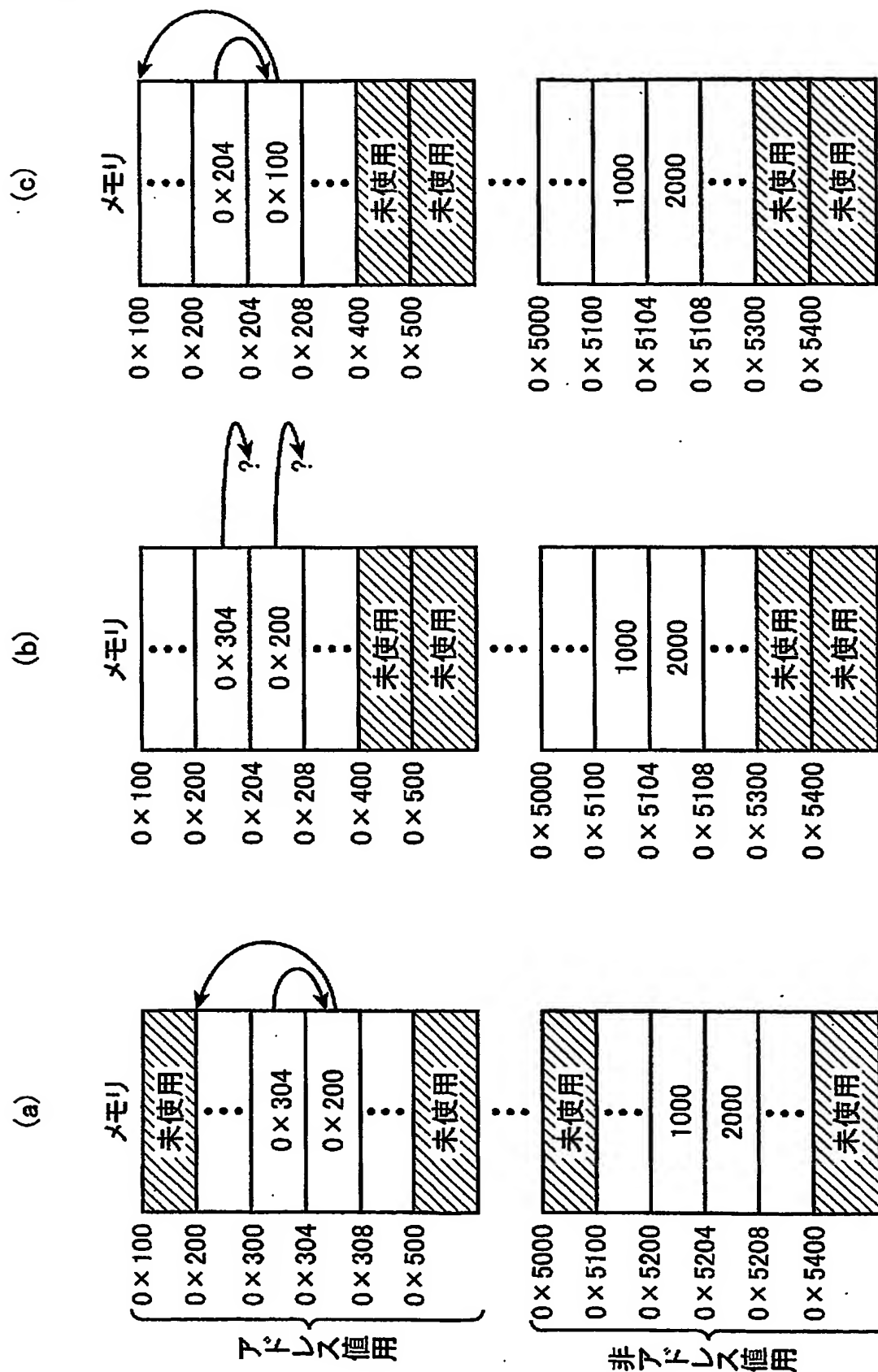
【図 23】



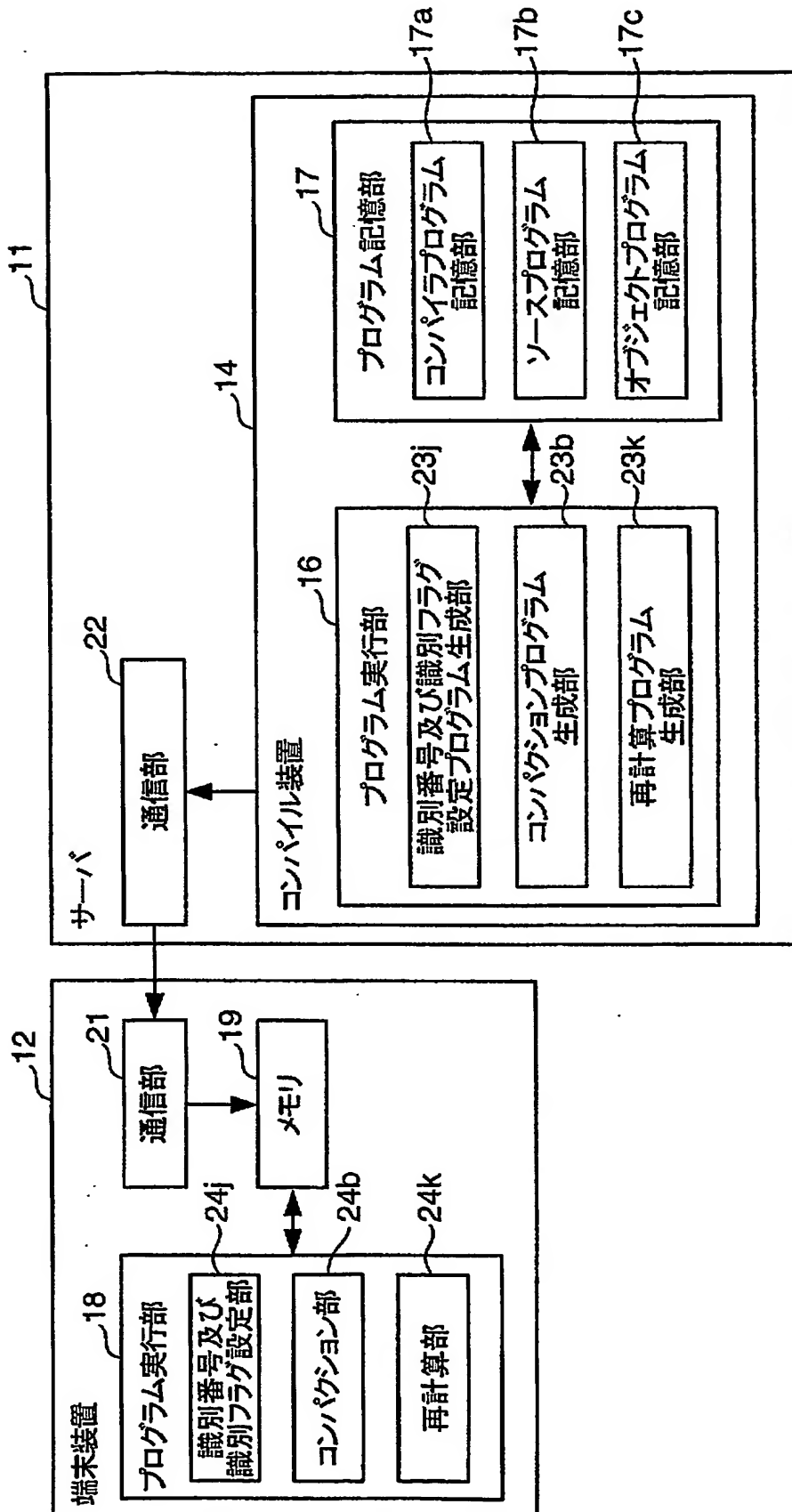
【図 24】



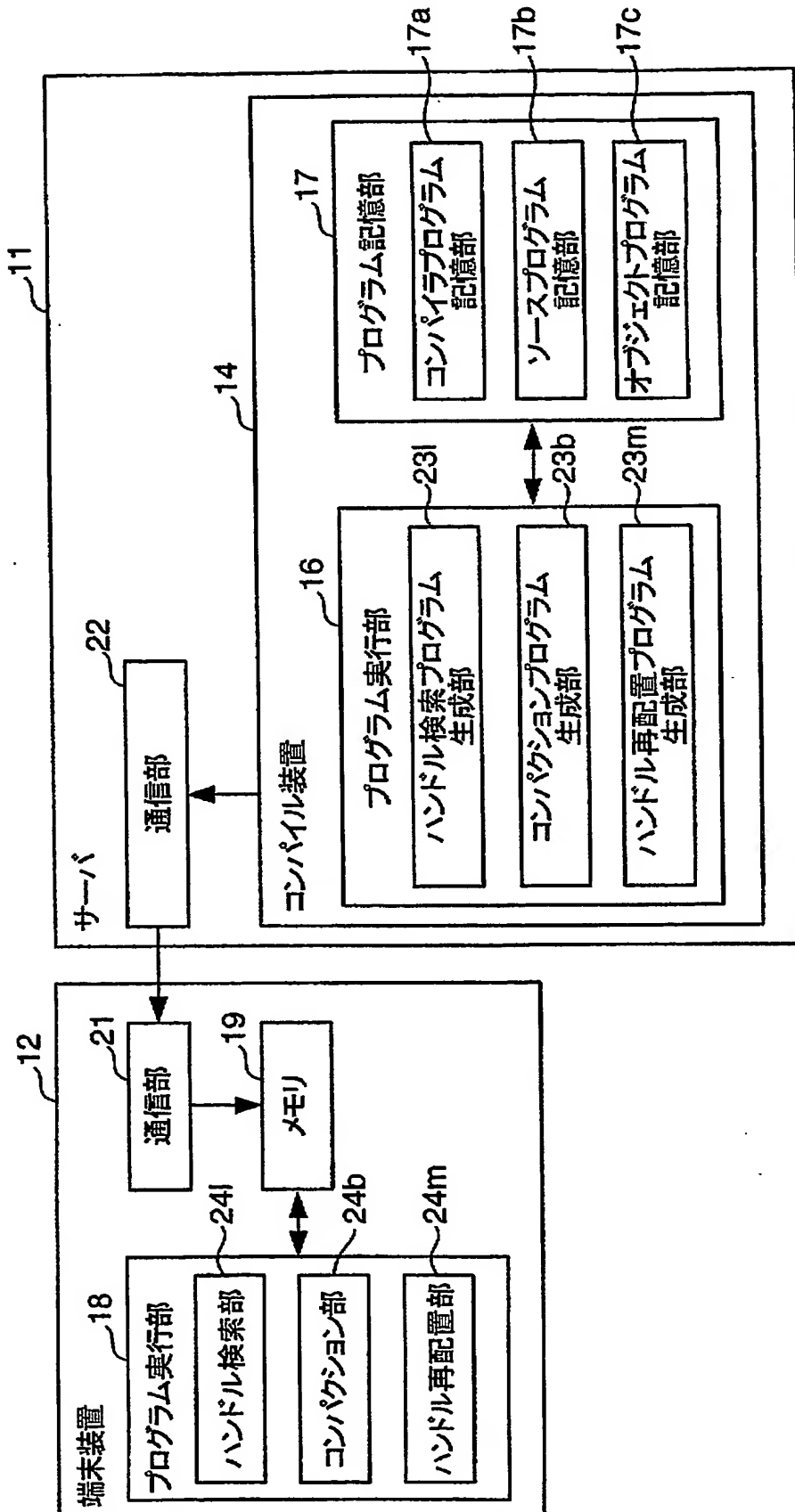
【図 25】



【図 26】



【図 27】



【書類名】 要約書**【要約】**

【課題】 自動的に再入可能な目的プログラムを生成する。

【解決手段】 アドレス保存プログラム生成部 16 a は呼び出し元プログラムモジュールのデータ領域アドレスを保存するためのアドレス保存プログラムを生成し、アドレス設定プログラム生成部 16 b は他プログラムモジュールのデータ領域アドレスを設定するためのアドレス設定プログラムを生成し、移行プログラム生成部 16 c は呼び出し元プログラムモジュールから他プログラムモジュールへ移行するための移行プログラムを生成し、アドレス再設定プログラム生成部 16 d は保存されたデータ領域アドレスを読み出して再設定するためのアドレス再設定プログラムを生成し、アクセスプログラム生成部 16 e は設定されたデータ領域アドレスからの相対アドレスで他プログラムモジュールのデータ領域にアクセスするためのアクセスプログラムを生成する。

【選択図】 図 2

特願 2003-402207

出 願 人 履 歴 情 報

識別番号

[000005821]

1. 変更年月日

1990年 8月28日

[変更理由]

新規登録

住 所

大阪府門真市大字門真1006番地

氏 名

松下電器産業株式会社